ENHANCING MULTISTAGE INTERCONNECTION NETWORK PERFORMANCE IN COMPUTER AND TELECOMMUNICATIONS SYSTEMS

BY

SANDRA E. CHEUNG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

1993

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Professor Yann-Hang Lee for his guidance, and infinite wisdom that made this work possible. His sound and thorough words have guided me in many aspects of academic life.

I would also like to thank my other committee members: Professor Y. C. Chow, Professor R. Newman-Wolfe, Professor T. Davis, and Professor S. X. Bai.

I am deeply indebted to my brother and friend Professor Shun Yan Cheung without whose encouragement and support throughout all of my life, but in particular during the past few years, this work would not have been possible.

A great number of people are responsible for instilling in me the passion to pursue knowledge and truth. I would like to thank all of my friends, colleagues, professors, and support staff who contributed to this experience. These include, but are not limited to, Dr. F. D. Anger, R. V. Rodriguez, Dr. M. E. Bermúdez, Dr. S. Sahni, G. Haskins, Javier, Padmashree, Jin-Joo, Balaji, and Mario. Special mention goes to the entire Happy Hour Gang for those sobering moments of truth.

I would also like to acknowledge my sister-in-law Stella, and my niece Nathanie for adding a new dimension to my life. Last, but certainly not least, I would like to thank my father 張福生 and my mother 巫惠英 for their love and vote of confidence, To them, I dedicate this work.

TABLE OF CONTENTS

CHAPTERS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree Doctor of Philosophy

ENHANCING MULTISTAGE INTERCONNECTION NETWORK PERFORMANCE IN COMPUTER AND TELECOMMUNICATIONS SYSTEMS

By

SANDRA E. CHEUNG

August 1993

Chairman: Dr. Yann-Hang Lee
Major Department: Computer and Information Sciences

Multistage interconnection networks (MINs) can provide an excellent tradeoff between cost and efficiency. Their regular structure and multiple access points make them not only amenable to cost effective VLSI implementation but also provide higher throughputs than bus based systems. Nowadays, MINs are used both in parallel computer and telecommunications systems. Though the design goals in these two areas are different, similar problems can be identified. Despite their advantages, the MINs exhibit *internal blocking*, which arises every time requests share similar links, even though they might be destined to distinct outputs. The blocking nature of these networks limits their throughput considerably, and is aggravated by the size of the network. In enhancing the throughput of MINs it is thus imperative to consider reducing the number of internal link conflicts. In computer systems we observe the effects of the *Consecutive Requests Traffic Pattern (CR)* in MINs. By identifying the causes for deterioration we are able to propose two methods, the *Dynamic Priority* and the *Bit Reverse* schemes which can significantly improve the performance under CR traffic. In telecommunications systems, the design goals are to grant as many connections as possible (in a circuit-switched design) and to pass as many packets as possible in a conflict-free manner (in a packet-switched design). We propose a new path setup scheme which can improve existing parallel path setup schemes. To enhance throughput of input queued fast packet switches, the *Circular Window Control* scheme can be implemented to schedule packets in a manner in which the probability of having conflicts is greatly reduced. The methods and schemes presented in this thesis show that with simple and cost effective enhancements, the performance of MINs can be enhanced considerably.

CHAPTER 1
INTRODUCTION

Communication bandwidth is a key factor in limiting the performance of not only high performance computers but also that of data communications networks. In both types of systems, multistage interconnection networks (MINs) provide the best tradeoff between cost and performance compared to other available means of interconnection. Some large scale shared memory multiprocessors such as the NYU Ultracomputer [GOTT83], the IBM RP3 [PFIS85a], and the Illinois Cedar [GAJS83] utilize MINs. The bandwidth of the interconnection between processors and memory modules, memory access delays, and the delays resulting from network and memory conflicts contribute to the overall system's performance.

In data communication networks MINs are used as a switch, i.e. a functional unit that receives units of information (sometimes referred to as cells) from a set of incoming channels and routes them as appropriate before transmitting the cells on a set of outgoing channels. With the advent of high transmission bandwidths, the main challenge remains to build switches which can handle these rates. One category of space division fast packet switches is based on multistage interconnection networks, in fact some of these include the Batcher-Banyan [HUI87] and the Starlite switch [HUAN84].

Common methods for optimizing MINs range from enhancing the switches of the MIN to scheduling the units of information in order to reduce the contention on the communication medium. Switch enhancement involves modifying the basic crossbar switches by adding circuitry that can handle additional functions. For example, by equipping switches with some bufferspace, conflicts can be tolerated in a more graceful way than having to drop the messages. Scheduling of data permits a much higher bandwidth to be achieved if conflicts can be actively avoided by not permitting them to access the medium at the same time.

The problem of improving the performance of a MIN can be addressed from several angles. In a computer system the network delay is a direct measure of performance and for this there are a number of ways to attack the problem:

- Reducing memory access delays.
- Supporting special traffic patterns.
- Improving bandwidth of the interconnect.

The efforts in data communication networks in turn, have concentrated on the problems which arose due to recent technologies. The accelerated development into higher speed communication has opened new avenues for all kinds of applications which in turn become dependent upon networks or networked computing applications. Building switches which can handle these enormous rates remains a challenge. MINs are used in the class of space-division fast packet switches and the methods for improving the performance of these switches aim at reducing or eliminating the *internal blocking* exhibited by MINs. Some of the common methods are:

- Reducing or eliminating head-of-line blocking.
- Scheduling requests in a conflict free manner.
- Providing multiple interconnection networks.

This thesis addresses the problem of providing a high degree of throughput using MINs in both high performance computer systems and in high speed communication networks. This thesis describes their vital role in both systems and presents efficient techniques for enhancing the performance of such switches according to the requirements.

Though the requirements of computer systems and communications systems differ vastly, a common goal of achieving an enhanced throughput can be obtained by studying the switch and its operations. In computer systems the switch can be modified in order to be robust under severe traffic conditions by supporting nonuniform reference patterns. In telecommunications' switches, throughput enhancement mechanisms vary according to the switching technique implemented. In a circuit switched architecture, the path setup phase should attempt to grant the most number of connections. In a packet switched architecture output contention and possibly internal blocking must be avoided by scheduling packets which will be non-contending. Often this may involve having to relax the strict FIFO order of the packet arrivals.

The first half of the thesis addresses the aspects of throughput improvement in computer systems, of which the primary emphasis is on the traffic patterns problem.

The remainder of this thesis addresses the problem of MIN-based switch enhancement in telecommunications. In here, the problem is examined both from a circuit-switching as well as a packet-switching point of view. This thesis further attests that these networks can provide excellent throughput and have applications in both computer and data communications systems.

The thesis is organized as follows. Chapter 2 gives a general overview of multistage interconnection networks and formulates the goals in both computer as well as in telecommunication networks. Chapter 3 details the background work and approaches taken in related work. Chapter 4 presents the Consecutive Requests traffic pattern and investigates the causes of network deterioration under this type of traffic. Chapter 5 studies the effects of the Consecutive Request traffic pattern when it interacts with regular traffic. Chapter 6 presents a parallel path setup scheme for connection oriented MIN-based high speed network switch. Chapter 7 presents the Circular Window Control scheme for scheduling packets in a highly nonconflicting manner in a MIN-based switch for high speed networks. Chapter 8 summarizes this thesis and discusses future research directions.

The methods presented are by no means exhaustive, and new techniques can be developed using the work described in this thesis.

CHAPTER 2
MINS IN COMPUTER AND TELECOMMUNICATION SYSTEMS

Multistage interconnection networks (MINs), and research associated with them, date back to the 1950s in connection with switching exchanges for telephone systems. In the early 1970s MINs were proposed for computer system applications and since then the research in both the areas of computer and telecommunications has reached numerous milestones.

With the present research on fabrication methods, there is no reason to suspect that the technological advancement will not continue. With the advent of high performance computers on one hand, and high speed networking on the other, MINs will be playing an integral part in providing data communication. The purpose of this thesis is to present methods to avoid or overcome pitfalls of MINs in computer and telecommunication systems.

The remainder of this chapter is organized as follows. An overview is provided in Section 2.1 to give the basic terminology and underlying topological design principles of MINs. Sections 2.2 and Section 2.3 deal with specific design issues of MINs in large-scale multiprocessor systems and telecommunication systems, respectively. Finally, Section 2.4 summarizes the requirements of MINs for computer and telecommunications systems.

2.1 Overview of Multistage Interconnection Networks

The vast array of physical implementations for interconnection switch ranges from bus-oriented to crossbar systems. The simplest of these is perhaps the time-shared bus, but it is unable to provide the performance required in a large-scale high performance system mainly because of its inability to support a large number of processors in terms of scalability. Moreover, buses are single access and would therefore would incur high contention. At the other end of the bandwidth spectrum is the full crossbar, which can connect any of its free input ports to any free output port by providing a separate switching gate for each input-output connection. A crossbar switch with N input and N output ports requires N^2 switching gates, and thus becomes the main drawback found with crossbar switches. The hardware costs required for interconnecting thousands of processors are too vast for crossbar switches to be considered economically viable. The feasibility of implementing large crossbars in VLSI presents another problem.

The main motivation behind selecting a multistage interconnection network as a communication medium is the hardware cost reduction. By connecting stages of smaller crossbar switches, a MIN can provide the required interconnection. While number of gates is significantly less than in crossbars $O(N \log_2 N)$, the speed is reduced because of the multiple number of stages and the time needed to set the network control, unless the network is self-routing (in which case the control is distributed to each switch).

In general, a MIN consists of a sequence of switching elements, arranged in stages. Physical wires connect successive stages and are also referred to as interstage links. The most common switching element used in a MIN is a crossbar switch, small enough to be economically attractive. For simplicity, we shall consider only networks with square crossbar switches, i.e. where the number of inputs is equal to the number of outputs.

MINs can be classified in a number of ways. For the purpose of this thesis, we shall consider one class of permutation networks (i.e. networks which can connect an input port to at most one output port) called *blocking* networks which exhibit the *Unique Path Property (UPP)*. As their name implies, these networks have a unique path between every input-output pair. However, a connection between a free input-output pair may not always be possible due to conflicts with existing connections, ergo the name *blocking*.

The network topology refers to the number of stages, the number and size of the switches, and the interconnection pattern between successive stages. Two networks are said to be topologically equivalent if their underlying graphs are isomorphic; they are isomorphic if there exists a label-preserving graph isomorphism between them. A topological equivalence relationship was established for several networks belonging to the class of UPP networks [WU80]. This equivalence implies they can perform the same set of permutation functions if the destinations are rearranged appropriately and that under the same routing algorithm, equivalent networks will have identical performance and fault tolerance characteristics. Equivalent networks can also be used to simulate each other.

UPP networks can be controlled with a distributed routing scheme, known as the destination tag algorithm proposed by Lawrie [LAWR75]. It is based on output labels called destination tags The binary representation of the destination is used to route the packet in stage i to the upper output port if $d_i = 0$ or to the lower port if $d_i = 1$ where the binary representation of the destination tag is $d_1 d_2 \ldots d_n$ (where n is the least significant bit).

An example of a UPP network is the Omega network [LAWR75]. An Omega network of size $N \times N$ ($N = 2^n$) consists of n stages. The network is composed of $log_2 N$ stages, where each stage contains $N/2$ switches and has a interstage pattern referred to as the perfect shuffle. An example of the Omega network constructed with 2×2 crossbar switches is shown in Figure 2.1, where N is equal to 8. In general, a multistage interconnection network can be built with a complexity of O($N \log_2 N$) in contrast to the O(N^2) complexity of a fully connected crossbar network. The destination tag routing algorithm is illustrated in Figure 2.1, where the bold lines indicate the route taken by a packet issued from processor $P1$ to memory module $M5$.

Due to the unique path nature of UPP networks, simultaneous connection of input ports to output port may lead to routing conflicts. The routing conflicts are due to the sharing of common internal links and are also referred to as internal blocking. Internal blocking can occur even though the ultimate destinations are distinct. In addition

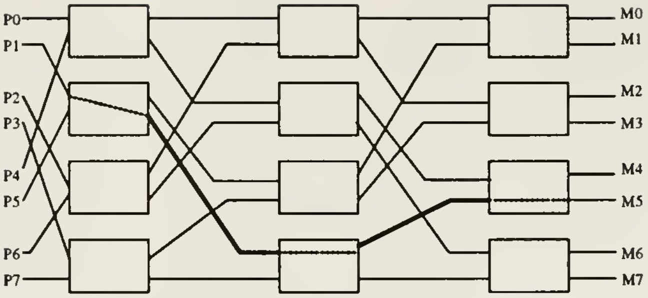

Figure 2.1. An 8 ×8 Omega Network

to internal conflicts, MINs also suffer from output conflicts, a type of blocking even crossbars experience. Output conflicts occur when multiple input ports request the same destination. Of the contending requests, one has to be delayed by one cycle. The victim is usually chosen in a round robin fashion. Conflicts may propagate back to the preceding stage and prohibit a successive message from transmission.

The regular structure of multistage interconnection networks makes it very amenable to VLSI implementation. In addition to this, its modular structure allows the construction of larger networks out of smaller ones without the need to substantially modify the physical layout or the algorithms.

The drawbacks to UPP networks are that the UPP leads to blocking and makes the full access capability more difficult in the event of a single failure. These shortcomings of the UPP networks have led to the design and implementation of enhanced UPP networks, or in multipath MINs. Another drawback of UPP networks is that a given UPP network cannot realize the full set of permutations from inputs to outputs without using multiple passes or by multiple copies of the network. Determining the minimum number of passes is not simple [PARK80]. In this thesis we shall not be concerned with faults and the discussion of enhanced networks will be the topic of the next chapter.

2.2 Multistage Interconnection Networks in Large-Scale Multiprocessor Systems

Network latency, compounded with memory access delay is known as the main factor limiting the performance of large-scale shared memory multiprocessors. Methods to reduce the memory access delay include memory hierarchies and asynchronous block transfers. Though I/O constitutes another important factor in multiprocessor performance, we assume that the applications under investigation are computationally intensive and that the I/O involved is negligible.

As computer technology has been moving towards high performance architectures, the trend has become to organize multiple processors (as much as thousands) which can then work in parallel on a single or on multiple tasks. These tasks are then partitioned into processes which are then assigned to different processors. Inevitably, these processes must communicate with one another and it is this cost which becomes a deciding factor in determining the performance and efficiency of the application.

Multistage interconnection networks have frequently been chosen to interconnect processors and memory modules due to many of its desirable features described in the previous section. Although there are a large number of different possible MINs, only a surprisingly small number are actually considered for multiprocessors. One that recurs in many different forms is the multistage shuffle-exchange network [BATC76, LAWR75, WU80].

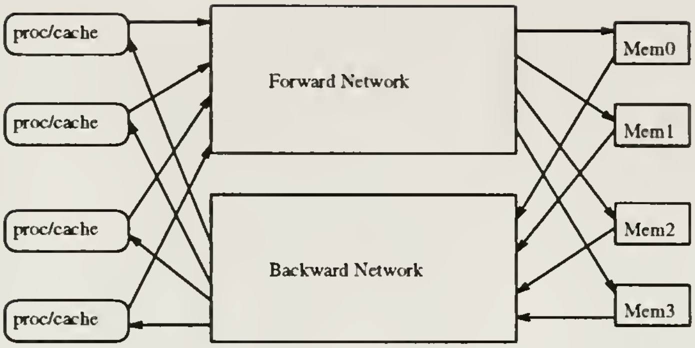

Figure 2.2. Large Scale Multiprocessor System Configuration

MIN systems together with memory interleaving can provide adequate network performance that not only depends on the memory system, but also on the rate at which processors issue memory references into the network (processor issue rate), which in turn is affected by the network delay.

Requests issued by a processor get buffered when they cannot be issued into the interconnect. After issuing a request, the processor waits for all the messages associated with that request to return. In general there are two types of messages that can be issued by a processor: *reads* and *writes*. Read requests are typically synchronous, i.e. the processor waits for the reply of memory in order to resume execution. However, in pipelined and vector processors, several reads can be issued in a sequence. Write requests, on the other hand, are asynchronous as processors do not need to wait for a reply. The requests which are initiated by the processors are sent through the forward network and memory modules return data through the backward network (Figure 2.2).

Network performance is not only dependent the request issue rate but also on the pattern which the requests form. The pattern of memory references generated by applications running in any computer environment constitute a traffic pattern. Some patterns have no particular structure to them and be considered quite random in their frequency and destinations. The reference pattern in which processing all memory modules with the same probability and where accesses are not dependent on one another is referred to as the *uniform* traffic pattern. The bandwidth of multistage interconnection networks can be very high for requests that do not conflict [LAWR75], but unfortunately most applications generate reference patterns which cause a high degree of conflicts in the MIN.

Nonuniform traffic patterns can be generated from a variety of applications which are data intensive or communication intensive. Nonuniform traffic patterns can cause a large amount of link contentions for an indefinite time and have a deteriorating effect on regular traffic as well. It is this effect that makes the study of traffic patterns which can arise from typical applications in multiprocessor environments important, for they directly impact the throughput provided by the interconnection network.

2.3 Multistage Interconnection Networks in Telecommunication Systems

Over the years, the growing number of applications which require high bandwidth has accelerated the research of high speed transmission media such as optical fibers. These applications are no longer solely computer related but can be found in areas such as video, voice and image, collectively known as *multimedia* [ARMB86, KIM90a, WOOD90]. The applications produce a range of traffic flow characteristics, which range in performance requirements. Not only must the transport network be able to guarantee these requirements but should also be flexible in offering bandwidth allocation, ensuring that each application type is allocated enough bandwidth to ensure its flow requirement.

Early networks were typically designed to support different traffic characteristics and requirements, and each was tailored for particular applications. Lately the drive is to design a single communication system which can provide the same services in a unified and integrated fashion. Some of the reasons are the ease of maintenance and installation, economy and ease of access. This movement has prompted the proposal of a set of services called *Integrated Services Digital Networks (ISDN)*.

The constant drive to achieve higher transmission bandwidth has led to the development of optical communications. The usage of optic fibers communication changed the requirements of data communication dramatically. Using lasers, which can be switched at a high rate, and optic fibers, the light signal can be carried over long distances with little attenuation. With this technology, data rates of 4Gb/s over 100 km without repeaters [TOBA90] are possible.

The emergence of numerous applications which require much higher bandwidth than possible in present networks was inevitable once the high transmission capacity of fiber optics technology was made available. The real challenge now becomes the creation of a network which can provide the high bandwidth services to the users. The bottleneck comes primarily from switching, since the data can be transmitted in a very high speed fashion. These high speed networks will carry all applications in an integrated fashion which all require different quality of service. The most appropriate switching technique is emerging to be packet switching which offers the flexibility to handle the wide diversity of data rate and latency requirements resulting from the integration of these services. Packet switching is also known as the asynchronous transfer mode (ATM) and circuit switching is known as the synchronous transfer mode (STM). At the present time, ATM specifies fixed-size packets of which 48 bytes are data and 5 bytes are control information. Line speeds which are specified are nominal rates of 150 Mb/s (for digitized high definition television (HDTV)) and 600 Mb/s [TOBA90].

Several architectures for fast packet switches have emerged in recent years, namely, (i) shared-memory, (ii) shared-medium, and (iii) space-division packet switches.

Of these three, space division fast packet switches allow multiple concurrent paths to be established from the inputs to the outputs. Space division switches can be categorized in two different types of fabrics: (i) crossbar fabrics and (ii) banyan-based fabrics. Though crossbar fabrics are nonblocking, the size of realizable crossbar switches tends to be limited. This is the primary reason why banyan-based fabrics have been considered as alternative candidates for fast packet switching.

The routing of data from an input to an output depends on the switching technique used in the switch. There are two principal switching techniques, circuit switching and packet switching. In circuit switching, a complete path of connected links from the origin to the destination is set up when the call is placed and before date is sent over it. The path remains dedicated to the call until it is released by the communicating parties. The overhead incurred during the setup phase is one of the cost factors which needs to be kept at a minimum. Circuit switching can be cost effective

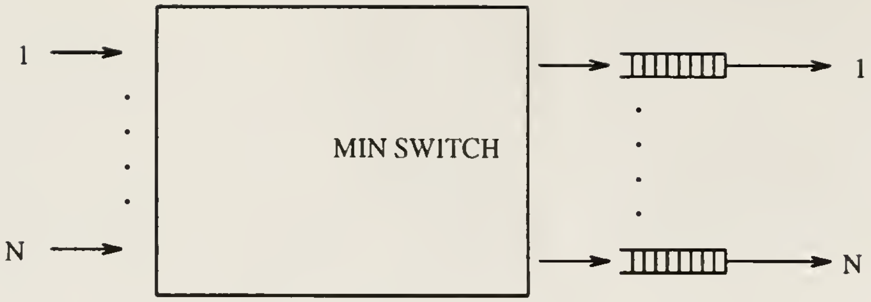

Figure 2.3. An Output Queued Packet Switch

when there is a steady flow of information and is hence the switching method used for voice communication.

Communication between computers tends to be bursty in nature, however. Circuit switching would be too costly and the circuits would be underutilized. Packet switching is a form of store-and-forward technique where the data is transmitted in chunks which are not to exceed a certain maximum length and are referred to as packets. Packet switching achieves benefits such as (i) dynamic bandwidth allocation, (ii) easy error recovery because smaller chunks are sent at a time. Once the packets arrive at their destination, reordering may need to take place.

Nonblocking MINs can be classified based on their queueing strategy: (i) input queueing, (ii) output queueing, and (iii) shared queueing. The speed of the switch fabric also plays a factor in where the queueing is done. For example, if in a $N \times N$ switch, the switch fabric runs N times as fast as the input and output links, all the packets that arrive during a slot can all be delivered at the outputs, even if multiple inputs request the same output. A slot is the time in which a packet can be transmitted. Output buffering is needed when packets arrived at the output port faster than the port itself can transmit (Figure 2.3).

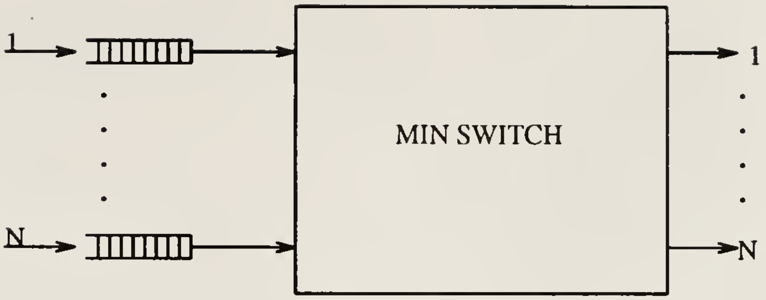

Figure 2.4. An Input Queued Packet Switch

Input queueing is necessary when each output can accept at most one packet per time slot (Figure 2.4). Input queued architectures have a maximum throughput of 0.586 [HLUC88] for an infinitely large switch, FIFO input queues with infinite queue length and uniform traffic. Thus, in spite of the queueing capability, its capacity is worse than an unbuffered crossbar switch. This phenomenon is due to the head-of-line (HOL) blocking. HOL blocking occurs when the packet at the head of the input queue cannot be transmitted and consequently blocks the other packets behind it although they may be addressed to idle outputs.

Typically there are three parameters used to describe the performance of the switching fabric:

- *Switch throughput.* The utilization of the output links is defined as the probability that a packet is transmitted in a time slot by the switch. The maximum throughput, also referred to as the switch capacity, is the traffic carried when packet arrival rate is one packet per time slot.

- *Average packet delay.* The average packet delay is defined to be the number of slots it takes for a packet received at the switch input to be transmitted to the addressed output.

- *Packet loss probability.* The packet loss probability is defined to be the probability that a packet received at the switch input is lost due to buffer overflow.

2.4 Chapter Summary

In this Chapter, a general overview of MINs was presented and found to be widely used both in large scale multiprocessor systems as well as in telecommunications systems. They provide a reasonable throughput at moderate hardware costs.

The requirements of the MIN depends highly on the environment in which it is being used. In multiprocessor systems, where the MIN is used to connect a number of processors to a number of memory modules, the objective is to minimize the network delay incurred by the memory references. These references are characterized by traffic patterns generated by applications running in the multiprocessor system. In telecommunications on the other hand, the MIN is used to switch data from incoming links to a outgoing links. The objective is to do this as fast as possible, with the least amount of internal blocking. Depending on the switching technique used in the switch, this requirement can be phrased as (i) improving the average number of setups granted (in circuit switched systems), and (ii) optimizing packet throughput (in packet switched systems).

CHAPTER 3
RELATED WORK

Most studies examined the behavior of MINs in isolation, and these do not reflect the characteristics of the entire system. The studies make simplifying assumptions, avoid the effects of head-of-line blocking, and thus obtain overestimating results.

The remainder of this chapter is organized as follows. Section 3.1 presents several studies performed on non-uniform traffic patterns and emphasizes the work done in modeling vector interference. Section 3.2 presents several setup algorithms previously proposed for some multistage interconnection network topologies and Section 3.3 details studies performed in improving the performance of banyan-based fast packet switches. Section 3.4 presents a summary of the related work discussed.

3.1 Non-Uniform Reference Patterns

One of the most unrealistic assumptions to make is that generated traffic always follows a uniform and independent pattern. The consequences of nonuniform traffic patterns, even when they occur sporadically, can have lasting effects on the entire system. This stresses the need to study the system in its entirety rather than studying its parts in isolation. Note that the non-uniform traffic patterns discussed in this section occur both in computer and telecommunication systems. The terminology is different, and we chose to use terminology used in a computer related environment.

Reference patterns in computer systems are the result of application programs executing and requiring communication to and from other processes, or from memories.

One type of reference pattern which has received much attention in the literature is the one in which one particular destination address is accessed with a higher probability than the remaining ones. This destination is also referred to as a hotspot. This phenomenon, first observed by Pfister and Norton [PFIS85b], causes the buffers of the paths leading to the hotspot to become saturated. The tree saturation phenomenon is a direct effect of hotspot access and leads to serious degradation in network throughput of not only accesses directed to the hotspot, but other memory references as well. The problem of hotspot contention has been studied extensively and various solutions have been proposed [PFIS85b, YEW87, HO89, SCOT89, DIAS89]. Hotspot traffic occurs also in communications networks, where it is referred to as output concentration.

When each processor references its own particular memory module more than others, the reference pattern which is generated is referred to as the preferred module, or in telecommunications' terminology, communities of interest. This type of traffic can occur in the case of bulk transfer. The hotspot traffic can be seen as an instance of preferred module traffic where all processors prefer the same module. The consequences of preferred module traffic are not as severe as those of hotspot traffic when the individual modules are well distributed. The traffic referencing a processor's preferred module obstructs other traffic from the same processor.

The burst issue rate is the rate at which the processor issues bursts of requests to the memory system. In most studies, the assumption is made that the burst issue rate is the same for all the processors in the system. A processor will typically wait for the burst requests to be satisfied before issuing the next burst. If the behavior has a distinct repeating pattern, it can also be termed as periodic. Burstiness is particularly of interest in communications, as burstiness is characterized by random gaps encountered during message transmission, variability of the message size and the low tolerance of delay of the source.

Another non-uniformity may occur when not all processors have the same issue rate. This case is also referred to as unbalanced input, and may not be of much interest in a computer system but is of utmost importance in telecommunications' switches.

3.1.1 Vector Interference

Numerical applications which operate primarily on vectors and matrices have a distinct pattern of reference. These systems are often required to use interleaved banks of memory. A parallel interleaved memory system allows concurrent access to multiple data items by placing these items in distinct memory modules. The assumption is that addresses can be presented to all the modules in parallel and that after a delay equal to the cycle time of the memory, data can be removed in parallel from all of the modules.

The Ultracomputer group [GOTT83] proposed to perform a hashing function which assigns each memory request a pseudo-random address, effectively eliminating any spatial locality. Gottlieb [GOTT83] states that hashing, however, only works for small systems.

In [CHEU86], a simulation study of the Cray X-MP's memory system was performed where the vector traffic caused linked conflicts. Vector activity was studied in a multiple bus architecture and suggestions were made with respect to the bank cycle time and the number of bus lines. Other studies on memory in vector supercomputers have been given in [BAIL87, OED85].

Storage schemes and address transformations have been proposed to reduce the contention created by the access of vectors. The conventional storage schemes are the interleaved scheme, and the skewed storage scheme [BUDN71]. The objective of storage schemes is to permit conflict-free access for a set of frequent access patterns. Storage schemes based on the access pattern of the vectors are presented in [HARP87, HARP89].

Vector traffic in a MIN and its adverse effects have been studied in [TURN89], where it was demonstrated that the forward network, memory, and backward network all affect each other. They propose placing long buffers at the memory inputs, which will eliminate blocking in the forward network.

Evaluation of the Cedar multiprocessor, which is a MIN-based system, was done in the work by [GALL91]. They observed that the degradation was due to the density of the requests which they resolved by adding dead cycles (NOPs) in order to reduce the density.

3.1.2 Vector-Scalar Interference

In many commercially available supercomputers, such as the Cray X-MP, multiple processors can access both vectors as well as scalars simultaneously. Their interaction can cause the degradation of one or both types of access types.

Processors which have a scalar cache cause cache blocks to be transferred between cache and main memory. Since data in cache lines are stored in contiguous sets of words, the interaction between these cache lines and ordinary vector accesses would result in the interaction between vectors, which was introduced in the section above. In order to observe vector-scalar interference, the system is assumed to be without any scalar caches.

In [RAGH91] the interference among vector and scalar accesses has been analyzed and found to reduce the performance of vector accesses that may already be in progress by as much as 40%. Their model assumed that the memory system had reached a conflict free steady state, operating at 100% efficiency. Other important assumptions are that there were no memory bank buffers present and that preference was given to scalar accesses. Then, a single scalar access caused a series of simultaneous bank conflicts leading to a detrimental effect on the memory efficiency. By using a crossbar interconnection network they avoided memory path conflicts which typically arise in MINs.

3.2 Path Setup Algorithms

An important aspect of interconnection networks is that of routing. A route specifies the links and resources which will be used to transfer the information from its source to its destination(s). Setting up a path between a source and a destination can be done using the self-routing property of a UPP.

Unique path property networks can realize only a small fraction of the $N!$ possible source-destination permutations. In order for a UPP to realize an arbitrary permutation, several passes through the UPP is necessary, where each pass realizes a submap of the permutation. The problem of finding the minimum number of passes to realize a permutation is intractable. Finding whether an arbitrary permutation can be realized in a given UPP topology has been done, for example for cube-connected networks in [ORUÇ91].

Generally, however, input patterns are not necessarily permutations. This adds another dimension to the setup problem. If n different sources request the same destination, at most one of these can send its packet.

A setup procedure is a mechanism which specifies which sources of a given request pattern get to send their data to the destinations. The setup procedure typically precedes the actual packet transfer protocols, particularly when the switching fabric is unbuffered and dropping of packets is inherent. In order to avoid retransmission, the setup phase ensures the packet can be transmitted prior to being admitted to the switch. The main goal of any setup procedure is to grant as many connections as possible at the least possible overhead. A subgoal in synchronous transmission is to combine this with synchronization that notifies the sources that the setup has terminated and that transmission may start.

The speed of the switch, with respect to the speed at which the input links are operating, has a direct impact on the path setup phase. If the transmission speed of the switch is n times as fast as the link transmission speed, then the packet at the input port has n opportunities to attempt to set up a path within a slot time. This increases the throughput considerably. Output queuing is necessary when multiple packets can arrive at the same output port.

The fundamental assumption made in most path setup schemes is that at the beginning of a transmission cycle, all setup requests are aligned and processed in a parallel fashion. Output link conflicts are usually resolved in a random manner whereby the victimized requests are dropped and denied a setup during the data transmission interval. Victors continue on to subsequent stages where again they might have output link conflicts. Ultimately, the victors of all stages are granted a connection. This is also known as parallel path setup (PPS), and notably Patel [PATE81] gives an analysis of the maximum throughput possible under these assumptions.

A study conducted by Lea [LEA92] showed that an incremental path setup (IPS) procedure can be implemented which can increase the number of possible setups at the expense of having more setup intervals (assuming the parallel path setup scheme only uses a single interval). Lea [LEA92] is able to achieve a spectrum of improvements by increasing the number of setup intervals. In order to achieve a reasonable improvement in throughput the number of setup intervals equals the number of contending sources (which is called pure IPS), the IPS scheme then attempts to setup each of the sources, one at a time. The order in which the setups occur (setup sequence) places a certain priority on the traffic at these inlets, the highest priority is a priori assigned to the inlet which is to be attempted first and the inlet at the end of the sequence has the lowest priority. Another drawback to the IPS scheme is that the data transmission cannot start until all the setup intervals are over, even though the network might be idle during some of these intervals due to a lack of requests (low input rate).

3.3 Fast Packet Scheduling

Most Asynchronous Transfer Mode (ATM) switches rely on the use of multiple stages of very simple switching elements, each of which is self-routing. MINs are constructed on these very principles and are thus heavily used in many of these proposals.

Blocking MINs require means of controlling packet loss by either buffering in the switching elements or by deflection routing. Rather, nonblocking MINs are used in ATM architectures. Nonblocking MINs can be further classified based on the queueing strategy adopted: (i) input queueing, (ii) output queueing and (iii) shared queueing. A nonblocking interconnection network guarantees the absence of internal conflicts and can thus establish a path from any free input to any free output.

3.3.1 Sorter Based Networks

A number of nonblocking MINs are based on placing a sorting network prior to a number of routing networks (this number depends on the speedup required). Both the sorting and the routing networks are multistage arrangements of sorting and switching elements respectively. An $N \times N$ Batcher sorting network has $n(n+1)/2$ stages (where $n = \log_2 N$), with each stage consisting of $\frac{N}{2}$ sorting elements. Additional means are required to guarantee that the sorted elements are all distinct and skewed into adjacent outlets.

It is well known that a routing banyan network is nonblocking if the set of packets to be routed is sorted based on the output addresses and received on adjacent inlets of the banyan network. Examples of sort-banyan fast packet switches are the Starlite [HUAN84], and others [HUI87].

3.3.2 Input Queue Architectures

Input queue switch architectures run at the same speed as the inputs and outputs, and packets are queued at the inputs. Typically, these input queues are serviced in a FIFO manner, creating the HOL blocking. This kind of blocking involves a packet at the head of the queue obstructing passage to packets behind it whose outputs are otherwise idle.

In the Three-Phase switch [HUI87] each slot is subdivided into three phases: *probe*, *acknowledgement*, and *data*. In the probe phase each active inlet issues a request packet indicating the outlet addressed by its head-of-line (HOL) packet. The interconnection network generates as many acknowledgements as there are granted requests and sends them back to their respective requesting inlets. All the inlets which have received the acknowledgement packet transmit their HOL packet during the data phase, through the sorting and routing network. Unacknowledged inlets try again during the next slot.

The Ring reservation switch [BING88] coordinates the inlets by interconnecting them in a ring structure. A reservation frame is serially transferred along the ring slot by slot so that each inlet can reserve the outlet addressed by its HOL packet, if it is not already reserved by some upstream inlet. Successful inlets transmit their HOL packet nonblockingly. The reservation phase and the data phase can be pipelined, but the serial reservation scheme makes the solution unviable for large switches, as the bit rate grows with the switch size.

The HOL blocking can be reduced in several ways [PATT88]:

- Switch expansion. The switch has more outlets (M) than inlets (N). The throughput is improved because the number of conflicts for each outlet is reduced.

- Windowing. Windowing is a technique that relieves the HOL blocking by allowing non-HOL packets to contend for switch outlets. A window of depth W means that a search is done for up to W packets, including the HOL packet.

- Channel grouping. The switch outlets are subdivided into groups of R each and packets now address groups rather than single outlets. In each slot each output link in a group is allocated to a specific inlet which is addressing that group with its HOL packet.

3.3.3 Shared Queue Architectures

The Starlite switch [HUAN84] is an example of a shared queue architecture. It uses a trap network interposed between the sorting and routing network. Its function is to mark and separate slot by slot the set of transmittable packets from the loser packets. The resulting sequence then consists of a sorted permutation, which after being skewed (removing the idle gaps in the sequence), is then passable in any UPP network. Winner packets are forwarded to the routing network and transmitted. Loser packets are fed back through a recirculation network to contend again with newly arrived packets. The recirculation network implements a distributed shared queue for packets that cannot be transmitted to their outputs.

3.4 Chapter Summary

In this chapter, relevant work in MINs was discussed and approaches taken by other researchers summarized. In the next chapters, we will present methods by which we can improve on the already presented techniques and/or solve problems which were also observed by others.

CHAPTER 4
THE CONSECUTIVE REQUESTS TRAFFIC PATTERN

Communication between processors in a shared memory multiprocessor is achieved by reading and writing to shared variables that reside in the shared memory. Each processor is typically equipped with its own local memory where its local data and its local program can reside. When these processors are executing in a parallel fashion, the computations should be arranged in such a manner so that the individual processors have to wait as little as possible. The traffic between the shared and local memories must be carefully managed since these data transfers incur a significant overhead.

The architecture of the interconnection affects parallel algorithm development and it is paramount to reduce the number of possible access collisions in this network. The performance of an interconnection network is dependent on the behavior of the imposed traffic. The pattern of the traffic is characterized by the rate of the requests and the frequency with which destinations are requested.

In a centralized-control SIMD environment, the network performance can attain its optimum when certain permutations appear in the memory access traffic [LAWR75]. Unfortunately, these special patterns cannot always be guaranteed in an environment with asynchronous executions on PEs. In most network performance analyses the common assumption is that all PEs issue their requests “independently” and "uniformly" to all memory modules. This means that at each source subsequent requests are independent from one another and that requests to memory modules are generated with equal probability. When the memory access traffic does not follow this independence and uniformity assumption, the performance is often overestimated.

Even if the memory accesses could be distributed uniformly, the independence assumption might not be valid. One instance is the consecutive memory access in which a request targeted to memory location i is followed by a request to location $i+1$. This kind of access pattern may occur when processors access a *stride-1* vector data or load program codes. This consecutive request (CR) pattern can also be caused by the transfer of a cache block between cache and shared memory if each processor is associated with a primary cache. When the interleaved memory allocation scheme is used, these consecutive memory accesses will be targeted to contiguous memory modules. In this case, the network receives sequences of dependent requests which may experience repeated network conflicts as observed in [SHEN89].

In vector applications, vector accesses are typically preceded and followed by periods of uniform traffic. These requests can be directed to synchronization variables or could be updates of local information. Hence our work considers the case in which at the time of the vector access the network might have some messages left from the preceding uniform access. These messages might obstruct the smooth passage of the CR traffic. Other factors which might affect the performance of our proposed schemes are the starting destination of the request sequences and whether all sequences were issued synchronously or asynchronously. We investigate these cases and evaluate the performance of the proposed solutions using simulations.

The evaluations show that, under the CR traffic, the performance of the original switch design is badly degraded and the solutions proposed significantly mitigate this degradation. Evaluations of the new designs are also done under the uniform traffic pattern to fulfill the nature of non-deterministic memory accesses in multiprocessor systems. These results show that the performance of the proposed schemes is the same as that of the primitive design running on the uniform traffic.

The remainder of this chapter is organized as follows. Section 4.1 discusses how, under the CR traffic, the network composed of the conventional switch elements will experience heavy conflicts in the initial stages. Section 4.2 presents two effective methods for mitigating the effects of the CR traffic. First, the Bit-Reverse address mapping scheme is proposed, in order to reduce the number of conflicts in the initial stages by reordering the destinations of continuous requests. Second, the Dynamic Priority scheme is proposed to increase the overlap of the uses of the two links at each switch. Section 4.3 discusses the effects of spatial distribution on the CR traffic pattern and generalizes the Bit-Reverse scheme. Sections 4.4 and 4.5 present the performance evaluation of the schemes on the forward network and backward network, respectively. Finally, Section 4.6 gives an overview of pertinent results obtained in this chapter.

4.1 Consecutive Request Traffic Pattern

It is questionable whether two successive requests from a processor are independent. If they are dependent, the question is whether switching elements would experience an unbalanced traffic and correlated conflicts. To investigate the impact of dependent requests, CR traffic pattern is described, which has the following characteristics:

1. A sequence of requests is generated at each source node and are destined to consecutive sink nodes, for example, the requests generated at processing element *PE1* to memory module *MM4* is followed by the requests to memory modules *MM5*, *MM6* and *MM7*.
2. This sequence of requests is generated continuously without interference from other requests.

The first place where this access pattern occurs is in vector accessing. A large number of numeric arrays and numerous iterations of computations involving array accesses are used in engineering and scientific applications. These data are distributed to many memory modules to prevent memory contention and increase computational parallelism. To execute these applications in a parallel manner, multiple processors are assigned to perform similar operations on a selected subset of operands. Furthermore, the use of instruction pipelining and data prefetching makes possible the prefetching of all these operands in very short intervals. Consequently, each processor will continuously request the data from memory modules with only one cycle in between. This traffic pattern is not solely restricted to vector processing. Cache block fetches and program loading have the same consecutive nature. In multiprocessor systems where each processor has its own private cache, or even in the event of a shared cache, upon a cache miss, the cache controller must read/write a line of words from/to global memory.

4.1.1 The Effects of the CR Pattern

In this section we will describe the effects of CR traffic in a network with conventional switches. Consider the activities in a switch in the first stage when the same CR sequence arrives simultaneously at each of its input ports. For example, *PE0* and *PE1* both send requests with destinations 000, 001, 010, ... , 111 consecutively in an 8 × 8 system. In the first half of the CR sequence, messages are addressed to the first half of the destinations, i.e., 000, 001, 010, 011. Therefore, the sequence of the first bits, used to control routing, remains the same; as in the above example, 0, 0, 0, 0. We define the duration in which routings remain unchanged as *Duration of Identical Routings* (DIR); in our example, the DIR at the first stage is 4. We can see that the possibility of insertion conflicts is high during this half of the CR sequence since the DIRs of both inputs are the same and long, i.e. both of the inputs try to route messages to the same output port for many cycles. The same thing occurs during the second half of the CR sequence which is directed to the lower half of memory.

In the second and the subsequent stages, conflicting messages will not only affect one of the messages at the input ports but also messages at the stages before it as well. We examined the network behavior and found that the CR traffic can severely deteriorate overall network performance for the following reasons. These are special cases of effects which will be generalized in Section 4.3. Below are the effects of CR traffic for unit stride accesses.

- *Effect (1')* The usages of the two links connected to the two output ports of a switch are not fully overlapped, i.e. resources are not fully utilized. As in the above example, at the output ports of a first-stage switch, the lower link is idle while the upper link is busy during the first half of the input CR sequences.

- *Effect (2')* The DIR is still long in the latter stages. If two input sequences overlap with each other, the length of the DIR has an adverse effect on the performance of the CR traffic.

If we can find a way that can prevent or reduce the above two phenomena, we can mitigate the performance degradation caused by the CR traffic. For simplicity, we shall call the above two effects created by the CR traffic as Effect (1'), and Effect (2') in the subsequent sections.

4.2 Dynamic Priority and Bit-Reverse Scheme

From the above, we can see that the network performance will degrade due to poor link utilization and long durations of identical routings. In this section, we propose two approaches to remedy these deficiencies. The approaches are easily incorporated in conventional switches and have the same performance as the conventional switch design under an independent and uniform traffic pattern.

4.2.1 Bit-Reverse Mapping

We consider the conventional switch design which uses the round robin priority scheme to resolve insertion conflicts. When a processor issues consecutive requests starting from the same memory module or approximately the same memory module, the destination addresses of these requests vary in low-order bits. However, the routing of messages in the first stage depends on the first bit of the destination address. Thus, the routes of successive requests will be overlapped in the first stages of the network. One way to split the routes of successive requests and to reduce their overlap is to adopt a bit-reverse mapping in the destination addresses and to rearrange the memory modules accordingly. In the interface between the processor and the network, we reverse the order of the destination addresses, for example, in an 8 × 8 MIN system, 3 (011) is mapped to 6 (110), and 4 (100) is mapped to 1 (001). The consecutive requests 0,1,2,3,4,5,6,7 are then mapped to 0,4,2,6,1,5,3,7. Correspondingly, the memory locations are also distributed according to the bit-reverse order in memory modules. By doing this, we have not changed the network performance of independent requests. It is the same as that without address mapping on uniform traffic since the destination addresses are generated randomly and are distributed to each destination with equal probability. No matter with which mapping function the addresses are mapped, they are still randomly distributed.

The Bit-Reverse mapping scheme can reduce the effect of CR traffic for a unit stride access because of the reduction of Effect (1') and Effect (2') in the first stage. After the Bit-Reverse mapping, the first bit sequence of the consecutive requests becomes a "0" followed by a "1" alternatively, so any two successive requests from a processor will have unoverlapped routes, except that they are received from the same input port. Also, if two consecutive requests enter a switch at the first stage, then the DIR is very short (only one cycle). If sequences arrive synchronously at the two input ports, a routing conflict arises. No conflict other than the initial one can occur in the first stage. As a result, both the output ports are kept busy forwarding data to the next stage. The usage of the two links from the same switch in the first stage is therefore well overlapped.

The main drawback to the Bit-Reverse mapping scheme is that Effect (1') becomes serious as the DIR increases in the latter stages. Since the round robin priority scheme is used to resolve insertion conflicts, all the messages that route to the same destination will cluster together at the stages approaching the destination. In other words, the blockage in the first stage of the network will be removed in order for requests to enter the network but these then get congested in the later stages. In spite of this deficiency, we expect that the advantages of the Bit-Reverse address mapping will lead to improvements in the network performance. This will be confirmed in Section 4.4.

4.2.2 Dynamic Priority Scheme

From the previous sections, we observed that when the round robin priority scheme is used to select which of the two conflicting messages is routed to the output port, Effect (1') and Effect (2') of the CR traffic could not be avoided. In order to reduce these effects we propose to use a dynamic priority scheme to resolve the insertion conflicts.

Adhering to the conventional switches with no address mapping mechanism at the processor-network interface, we will now give preference to a message coming from a particular input port in the cases when successive input messages are trying to route to the same output port.

In Dynamic Priority any input port can be initially given the priority and it will keep this priority as long as it continues having messages in its input register in subsequent cycles. Upon resolution of insertion conflicts the priority circuit will indicate the register which has the *current* priority. If at any point in time, the input register holding the priority has no message but the other one does, the priority circuit will revert to reflect this change of traffic.

Since the insertion priority is given to a particular input, if the next arriving message is to be routed to the same output port, the first half of the CR sequence from that input can pass quickly without being blocked. Therefore, the second half of the CR sequence can appear at that input earlier and also pass through the second output link of the switch earlier. That is, the overlapping of the link usages from the same switch is higher and Effect (1') is reduced. Moreover, since the sequence of messages to the second and subsequent stages remains in a consecutive order (i.e. not mixed with the sequence issued by other processors), the DIR length is halved in these stages. In our example, the sequence remains 000, 001, 010, 011 when it arrives at the second stage and the DIR at this stage is two. So, Effect (2') can be reduced in the second stage and subsequent stages.

An intuitive drawback of this approach is that the unfair message-passing would cause uneven responses to the processors. Therefore, unfair services would be perceived by the processors. However, if all the processors in the network are performing vector accesses, this unfairness within each switch element does not have an impact on the performance of the overall system. Also, in uniform traffic, this unfairness is not present mainly because the DIR of any stage in this type of traffic is on the average two.

4.2.3 The Combined Approach

The hybrid model which combines both the Dynamic Priority scheme as well as the Bit Reverse mapping offers the same benefits as the ones described above, but with one added benefit. Observe that the Dynamic Priority allows for rapid passage of messages in the early stages but in the latter stages it is still almost round robin since the DIR in this stage is not long. In Bit Reverse mapping however, the DIR becomes larger in the last few stages and the Dynamic Priority could alleviate this in the same manner as described, but effective in the last stages, where the Bit Reverse mapping needs it most. We therefore expect this scheme to have the advantages of both.

4.3 Effects of Spatial Distribution on CR Pattern

There are two aspects to the spatial distribution of a CR traffic pattern: the bank offset and the stride. The bank offset is the starting position of the vector access which can be of paramount importance to the performance of the memory access. The stride of an access refers to the constant offset between consecutive requests. For example, a processor which starts at module 0 and accesses with a stride of 2 will send the requests to the even numbered memory banks. Hence it can be seen that the stride not only affects the order in which memory modules are accessed, but also the number of distinct memory modules which are accessed.

Bank offset can be an important factor in the performance of CR accesses. The effects can be seen from observing that the DIR of the sequences depends highly on the sequences which arrive at the input ports of a switch. The most restrictive case is when both the sequences are identical, which makes the DIRs at both inlets of the same length and cause the least amount of overlap. When the sequences differ by just one module, it creates some overlap and reduces the Effect (1') mentioned above.

In some vector computers the stride is an important parameter. Some vector computers (the Cyber-205, for instance) allow only a stride of 1. These types of processors use "gather" and "scatter" methods to create temporary vectors which are contiguously stored. Most vector computers can process strides other than 1, and in our processor model we assume the processors can handle strides larger than 1.

Stride accesses other than 1 are not unusual. In fact, a number of numeric array accesses and numeric computations in loops involving array accesses contain loop iteration variables which are incremented by numbers other than 1. In Figure 4.1 a code segment is given where the participating processors are accessing every other element of a vector A.

```
DOALL i=1, number-of-processors
      DO j=0, number-of-vectorelements, 2
      // step size is 2 between consecutive iterations //
                 LOAD A(j)
      ENDDO
ENDDOALL
```

Figure 4.1. Stride-2 Access

In order to see the effect of the stride on a network with conventional switches we will assume we have the same type of scenario as described above. We will consider the activities in a switch in the first stage when the following CR sequence arrives simultaneously at each of its input ports. Assuming the stride to be 2 and the access to start from memory module 0, we then observe that at a first stage switch, the sequence with destinations 000, 010, 100, 110, 000, 010, 100, 110 consecutively arriving in an 8 × 8 system. It can immediately be observed that only the even numbered modules are addressed. The DIR of this sequence for the first stage is 2, half of what it would have been in the case of a stride-1 access. The possibility of insertion conflicts is still high due to the length of the DIR at both inputs (this is particularly the case for larger systems).

The next observation to be made is that the DIR for the second stage is 1. This is because the stride-2 access causes the second least significant bit to toggle. This is consistent with the findings of the unit stride access example in Section 4.2.1, in which it was seen that the least significant bit made the DIR of the last stage equal to 1.

In general, a stride-i access can deteriorate network performance for the following reasons: (the first two are a reformulation of the two effects stated in Section 4.1).

- *Effect (1)* The usages of the two links connected to the two output ports of a switch are not fully overlapped, i.e. resources are not fully utilized. In essence, this underutilization may occur at different stages, depending on the stride, and the size of the network.

- *Effect (2)* The length of the DIR depends on the stride as well as on the size of the network and is long while traversing stages which do not use the bit toggled by the stride (henceforth referred to as the *stride-bit*). In the stage which uses the stride-bit, the DIR is exactly one, and doubles with each subsequent stage. If two input sequences overlap with each other, the length of the DIR has an adverse effect on the performance of the CR traffic.

- *Effect (3)* A stride-i access is in essence a special kind of multiple hot-spot pattern since only $\frac{n}{i}$ memory modules will be accessed (assuming there are n memory modules and that i divides n evenly). In the worst case, all the traffic is directed to only those modules.

Strides which are not powers of 2 do not have the deteriorating characteristics which are described by the effects above. The primary reason for this is that the length of the DIR of such accesses is not as long as for accesses of powers of 2. We will therefore assume that the strides in this study are all powers of 2.

4.3.1 Dynamic Priority and Stride Based Mapping Revisited

The Dynamic Priority scheme as described above can still be effective with stride accesses other than 1. It is well understood that the most restrictive case is the stride-1 access (where the DIR is the longest) and hence that type of access can benefit the most from the Dynamic Priority. However, Dynamic Priority still offers the same benefits to access of other strides, though the underutilization is perceivably less, due to their shorter DIRs.

The Bit-Reverse mapping scheme described in Section 4.2.1 did not take the stride into consideration, but rather assumed it to be 1. The reason why this approach is not sufficient here is that this method was meant to spread the accesses of a stride-1 access which accessed all the modules in a consecutive fashion. Strides other than 1 cannot take advantage of this characteristic and hence a new mapping is proposed which generalizes the Bit-Reverse mapping previously described.

We will show by means of a simple example why the original Bit-Reverse mapping is not adequate for strides other than 1. Suppose we had the sequence of requests: 000, 010, 100, 110 (access stride is 2). Using the original Bit-Reverse mapping scheme would map these to addresses 000, 010, 001, 011 respectively. These newly mapped addresses all lie in the lower half of the memory modules. With higher numbered strides (powers of 2), the mappings will concentrate in an even smaller corner of the address spectrum.

When a sequence of requests are issued with a stride of i, the destination addresses will toggle in bit position $d_i = \log_2 i$, assuming the addresses are represented as

$d_{n-1}d_{n-2}\ldots d_0$, where d_0 is the least significant bit. The original motivation of the Bit-Reverse scheme was to split the routes of successive requests in order to allow more overlap in the initial stages of the MIN. By applying the Bit-Reverse mapping, the DIR of successive requests became 1 (in the first stage) and gradually grew as the stages approached memory. In order to achieve the same results with the sequence generated by a stride of i, the reversal of bits should involve only bits $d_{n-1}d_{n-2}\ldots d_i$ and the remaining bits $d_{i-1}d_{i-2}\ldots d_0$ should stay the same. This is the general stride based mapping family of which the Bit-Reverse is a particular case (for stride 1).

Using this stride based mapping scheme the sequence 000, 010, 100, 110 gets mapped to 000, 100, 010, 110. Compared to the mappings created by the Bit-Reverse mapping, the newly acquired destinations are scattered in the upper half as well as the lower half of the memory modules, instead of concentrated in a corner of the address spectrum. By distributing the addresses, the network performance of independent requests has not changed. It is still the same as that without address mapping by the same argument as was given in the Bit-Reverse discussion.

The general stride based scheme can reduce the effect of any CR pattern because it reduces both Effect (1) and Effect (2) described above. The drawback of the scheme is still that the DIR increases in the latter stages. Moreover, Effect (3) has not been affected by the mapping scheme. We shall see later that Effect (3) becomes the predominant reason that stride based mapping alone is not sufficient, particularly as the stride increases.

4.3.2 Skewed Storage Schemes

Interleaving memory works well when reference sequences address contiguous memory modules. Strides other than one are quite common in a multiprocessor vector processing environment where each processor will generate highly correlated address sequences [HARP87]. The interleaved system will suffer under strides other than one because references will not be distributed to all memory modules.

Using any of the stride dependent schemes described above will not increase the number of distinct memory modules being accessed (Effect (3)). Skewing schemes have been used to eliminate the conflicts which arise from parallel access due to strides greater than one. It has been proven [BUDN71] that there is no single skewing scheme which allows conflict free access to a vector using all strides. There are several classes of skewing schemes [SHAP78, WIJS85] but the scheme which is used in our experiments is defined by a function which maps the ith element of a vector to memory module $\left(i + \left\lfloor \frac{i}{N} \right\rfloor\right)$ mod N where N is the number of memory modules and $\lfloor i/N \rfloor$ denotes the largest integer less than or equal to i/N. Table 4.1 illustrates the skewing scheme in an eight module system.

In order to see the relationship between the stride (S) and the number of memory modules (M), we consider the case in which both the stride and the number of memories are powers of 2 ($S = 2^s$ and $M = 2^m$). We will consider two cases, strides in which there is at least one access per row (for $S \leq M$ or $s \leq m$), and strides in which there is less than one access per row (for $S > M$ or $s > m$).

For the following discussion, we assume that the initial access of the vector is to memory module 0. There is no loss of generality if we assume this. Assuming an interleaved memory scheme, the sequence of references is: $0, 2^s, \cdots, 2^{m-1}$. After 2^{m-s} references the sequence repeats. In the interleaved scheme, the $2^{m-s}th$ access falls in the same module as the initial access. This obviously causes a collision with the initial access. However, if the current row is skewed relative to the former row, then rather than referencing the same modules as the former row, references are made to modules adjacent to the corresponding modules in the former row. In other words, to allow M accesses to occur before a conflict occurs, it is necessary to skew 2^s rows relative to each other. For any stride $S = 2^s$, with $s \leq m$, a skewed storage scheme would consist of circularly rotating reach row r by r mod 2^s places relative to its original location in the interleaved scheme.

In the case where there is at most one access per row, the sequence of references is $0, 2^s, 2 \times 2^s, \cdots$. All references will fall in the same module which causes the most performance degradation. Rotating the rows can alleviate this but the pattern is different than described above: blocks of contiguous rows are rotated relative to preceding blocks. Blocks of 2^{s-m} rows are rotated in unison, which is precisely the number of rows between consecutive elements. The rotation places consecutive accesses in adjacent modules. Over a period of M accesses, each module is referenced exactly once.

In general, a skewed storage scheme is based on two parameters:

M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	1	2	3	4	5	6	7
15	8	9	10	11	12	13	14
22	23	16	17	18	19	20	21
29	30	31	24	25	26	27	28
36	37	38	39	32	33	34	35
43	44	45	46	47	40	41	42
50	51	52	53	54	55	48	49
57	58	59	60	61	62	63	56

Table 4.1. Skewed Storage Scheme

1. The maximum rotation a row can have relative to the first row (r_{max}). For $s \geq m$, r_{max} is always M, and for the $s < m$ case, r_{max} is given by 2^s.

$$r_{max} = min(2^s, M) \tag{4.1}$$

2. Amount of memory to be rotated as a single block. In the $s < m$ case, this is always 1, otherwise it is equal to 2^{s-m}.

The efficiency of any skewed storage scheme is measured in terms of its ease in generating addresses, and the number of possible conflict free accesses it can provide (a parallel access is said to be conflict free if it can be accessed in a single memory cycle, i.e. without conflicts).

In the context of a multistage interconnection network, a skewed storage scheme will improve the performance of a CR traffic pattern, mainly because Effect (3) above is alleviated. However, note that there is still underutilization of the links since the pattern is still distinctly a CR pattern. For example, suppose we had a stride 2 access which in an interleaved storage would yield the following access pattern: 000, 010, 100, 110, 000, 010, 100, 110. In a skewed memory storage the reference pattern would then be: 000, 010, 100, 110, 001, 011, 101, 111. It can be seen from this simple example that the output links will be underutilized in certain portions of the sequence. When we apply the stride dependent Bit-Reverse mapping, we get the sequence: 000, 100, 010, 110, 001, 101, 011, 111 (or 0,4,2,6,1,5,3,7). Interestingly enough this is the same pattern which we would get when applying the Bit Reverse mapping scheme onto a unit stride access starting at module 000. From the above we will then expect similar performance of a Bit-Reverse mapping scheme of a stride-1 access in an interleaved memory and a stride dependent mapping scheme of stride-2 in a skewed memory.

In our investigations, we assume that a vector is only accessed with a single stride. Even if vectors are not always accessed with a single stride, it is by far the most common case. An example in which a vector is not accessed with single stride is the well known row/column access pattern found in matrix manipulation routines.

4.4 Performance Evaluation of Forward Network

In order to measure the performance of the system under CR traffic and to evaluate the effectiveness of our schemes, we have conducted a simulation study of the Dynamic Priority and Bit-Reverse (and stride based) schemes in a multistage interconnection network.

The simulation model used a 64 × 64 buffered Omega network. A larger network has been used in additional studies and have yielded similar results. The total message capacity of the queues at the output ports is varied from four, six to eight messages each. The reason the size of eight was chosen is that in [KRUS83] it was shown that a queue size of eight could model infinite queue size, so that blockage caused by a full queue can be almost eliminated. A processor has at most one outstanding request and stops issuing a request when the corresponding input register of the first stage of the MIN is busy. Once the processor starts issuing CR messages, it will generate one every clock cycle, if it can insert it into the network. This dependency of subsequent messages is the main characteristic of the CR traffic. The time at which the processors start their vector access is also varied, from synchronous to asynchronous.

To study the spatial effect caused by the bank offset, we shall distinguish between two types of access, CR sequences starting with the same bank offset, and CR sequences access with bank offsets chosen randomly for each processor. The spatial effect caused by different stride accesses is investigated in conjuction with the stride-dependent mapping and skewing schemes. Finally, we also study vector accesses under an idle network before the access, and under an initial load of uniform accesses.

The code which implements the CR concurrent fetch is typical (Figure 4.2). Each processor is assumed to run the exact same code, and this code is assumed to be pre-scheduled (to minimize the effects of scheduling overhead). We assume that a single vector load is made, and that no NOPs are inserted between fetches.

The main measures which we took from simulations using the CR traffic model were the average message delay, which is defined to the number of cycles it takes

```
DOALL i=1, number-of-processors
      DO j=1, number-of-vectorelements
                 LOAD A(j)
      ENDDO
ENDDOALL
```

Figure 4.2. CR Concurrent Fetch

on the average for each message of the vector sequence. the throughput (number of messages which reach memory per cycle), and the average sequence (vector) delay, which is defined as the average number of cycles it takes for all the processors to complete their vector access in the forward network. It represents the average of all the worst case completions over all iterations.

These measurements conform to a processing model in which a number of processors, *N*, participate in a barrier synchronization. As processors reach the barrier they are suspended and wait until the last one finishes. The barrier is the most restricted form of synchronization. Thus a good measure of performance is the worst case execution of each of the processors. This is an indication of how long the other processors must wait before they can proceed. In our model, the execution is the issuing of a CR sequence and the measuring of the average sequence delay, as defined above, is a measure of the average time the processors need to spend during the barrier synchronization.

The simulation results presented in this section are based on a single vector access of equal length, issued by all the processors in the system. Except for the synchronous

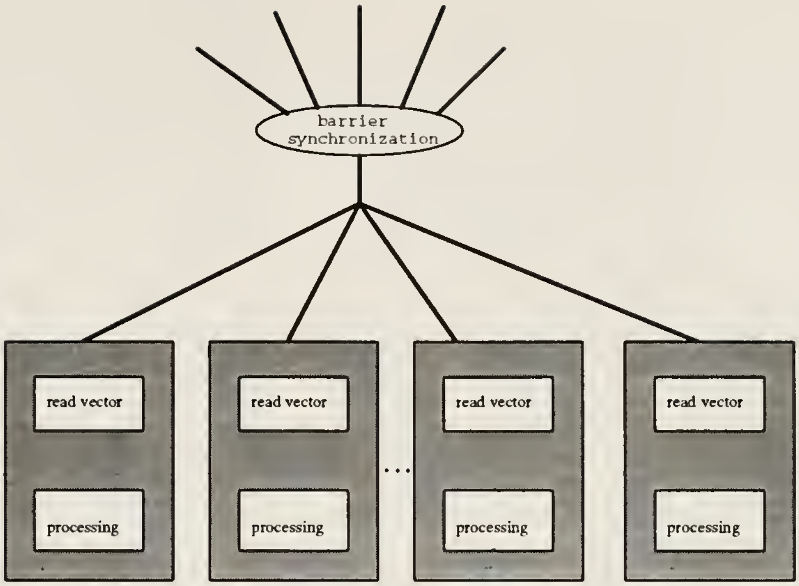

Figure 4.3. Processing Model

case, a number of 200 runs were collected per simulation point (measurements were taken at interstart periods of 0.0, 5.0, 10.0, 15.0, and 20.0). The comparison to the optimum is also given. The optimum always assumes that there are no conflicts in the network. The model for this section assumed no memory model, i.e., that memory modules had infinite length input buffers. The optimum vector delay in every graph, is also provided to illustrate the point and the illustration speaks for itself. The optimum vector delay is one in which it is assumed that there are no conflicts in the network.

4.4.1 Dynamic Priority and Bit-Reverse Mapping Results

The results presented in this section assume the CR sequence to be accessed with unit stride. Our simulation primarily focused on the relative performance of the conventional switch which uses the round robin strategy and our proposed schemes, the Dynamic Priority, Bit Reverse mapping, and their combination. In the discussion

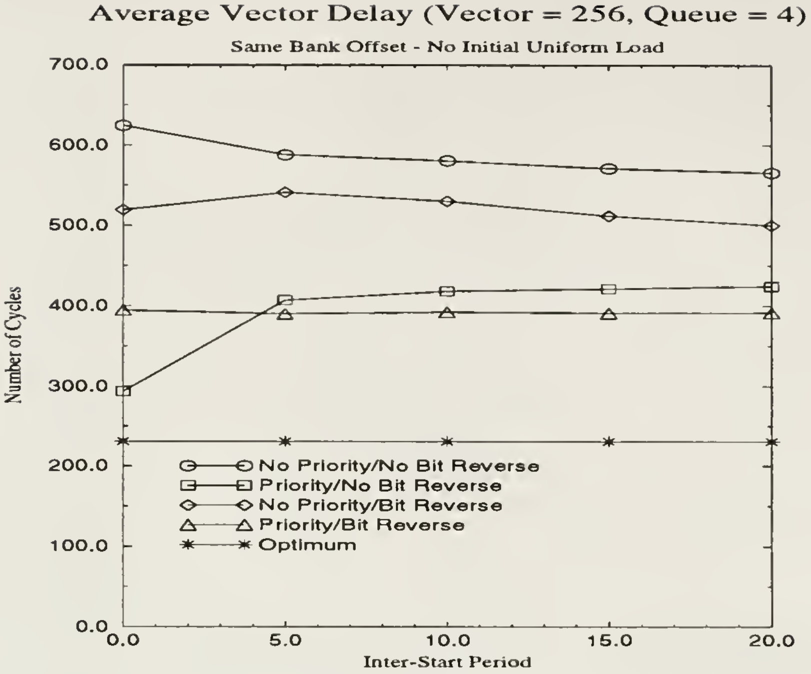

Figure 4.4. Same Bank Offset/Stride 1 - No Initial Load

which follows we will refer to the round robin strategy with no memory mapping as the conventional design. Measurements were collected for the conventional design, the Dynamic Priority, Bit-Reverse, and the hybrid scheme. The time that each processor begins to issue its CR requests is distributed within the inter-start period.

The simplest scenario is the one in which there is no initial load on the system and the processors issue CR sequences with the same bank offset. Figure 4.4 shows the average vector delay for these four schemes under these assumptions. From this figure we can conclude that the Dynamic Priority scheme gives the best results if the processors start synchronously. In the asynchronous case this scheme deteriorates but the combined scheme performs slightly better than the Dynamic Priority alone. It is interesting to observe that the combined approach has a rather stable behavior.

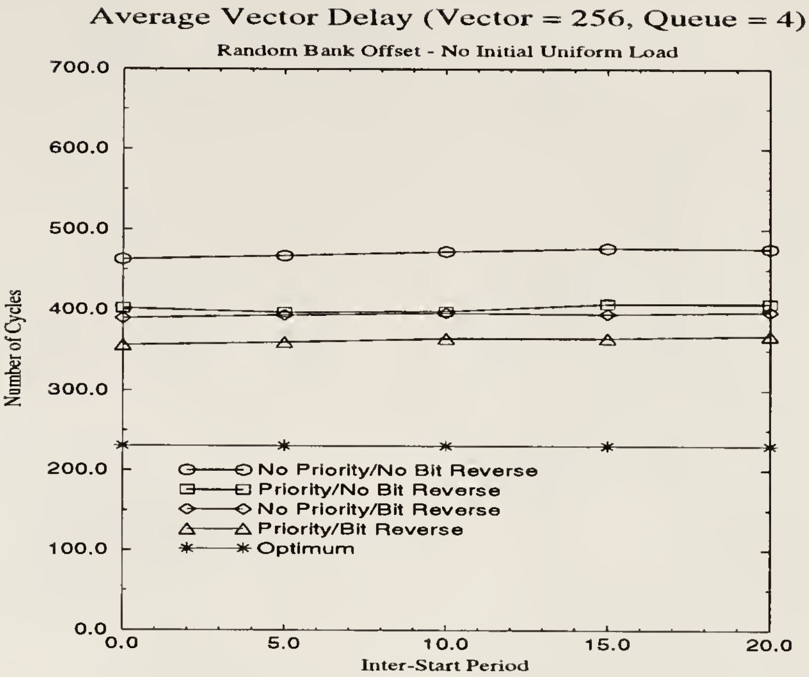

Figure 4.5. Random Bank Offset/Stride 1 - No Initial Load

In all cases our proposed schemes yield better performance than the conventional design.

It is of importance in which sequence the allocated vector is being accessed since this determines the sequence of requests issued by a processor and also the number of conflicts. Our simulation has concentrated on CR sequences with the same bank offset and for completeness we will show how our schemes would perform under access to different memory starting modules. This type of access causes fewer conflicts and its performance can be seen in Figure 4.5. In this case, the hybrid scheme yields the best results while the individual schemes perform alike. It can also be seen that all the proposed schemes outperform the conventional switch design.

Processors usually do a series of random accesses before and after performing vector accesses. The results of the four schemes with CR sequences with the same

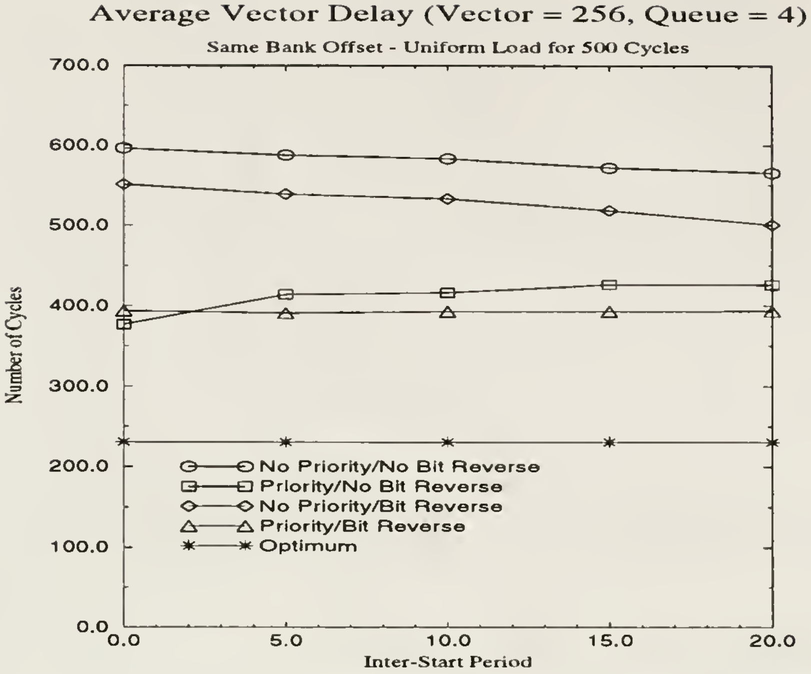

Figure 4.6. Same Bank Offset/Stride 1 - Uniform Load for 500 Cycles

bank offset access where the CR traffic is preceded by a uniform traffic period of 500 cycles at a request rate of 0.5 is shown in Figure 4.6. The initial load has a larger effect on the performance of the synchronous Dynamic Priority scheme, since this scheme is more vulnerable to the network being idle before the access. The hybrid scheme shows the most stability, as can be observed from Figure 4.4 and 4.6 that there is no change in the curve. We also obtained the case in which the same initial load precedes a CR access starting from different bank offsets. The results are presented in Figure 4.7. Remarkably, the initial load has no effect on the average vector delay in the case of an access with different bank offsets. The reason for this is that the initial load makes the DIR smaller by interfering with the vector access.

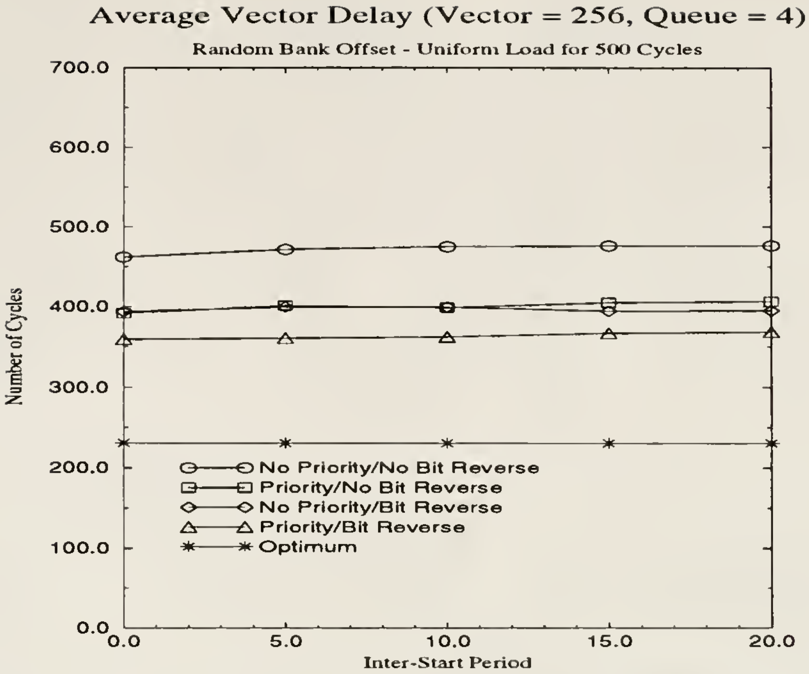

Figure 4.7. Random Bank Offset/Stride 1 - Uniform Load for 500 Cycles

Another factor which we measured is the throughput under the CR traffic. From Figure 4.8 it can be seen that the throughput obtained under synchronous access using the Dynamic Priority scheme is the highest.

Figures 4.8 and 4.9 show an interesting property of the Dynamic Priority scheme under synchronous and CR sequences with the same bank offset access, that irrespective of the length of the vector, after an initial setup time (different for each processor), each processor will deliver messages with the same message delay. Once the first request reaches memory, they keep arriving at a constant rate of one per cycle. This indicates that after the bottleneck is resolved in the early stages, the pipeline of requests is operating to its fullest capacity.

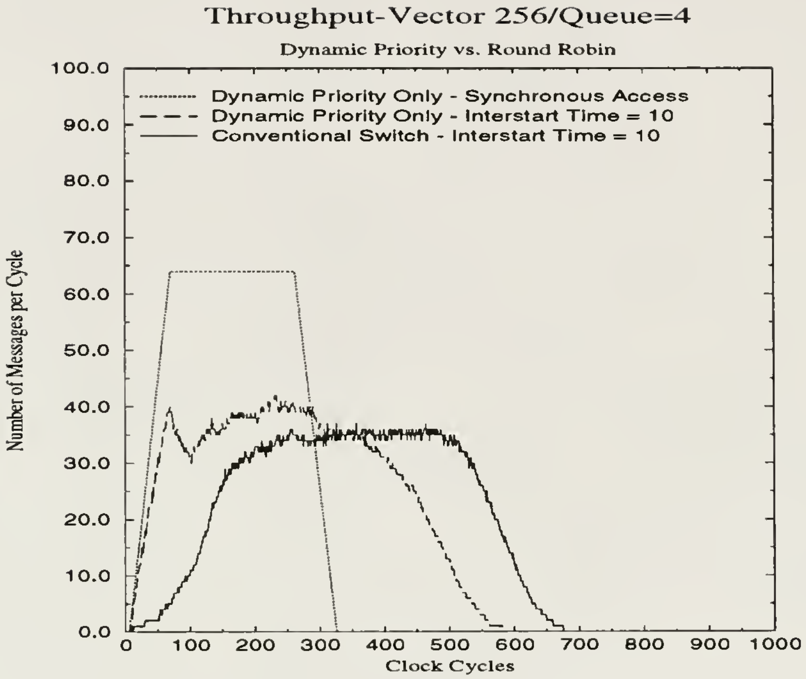

Figure 4.8. Throughput for Dynamic Priority and Round Robin

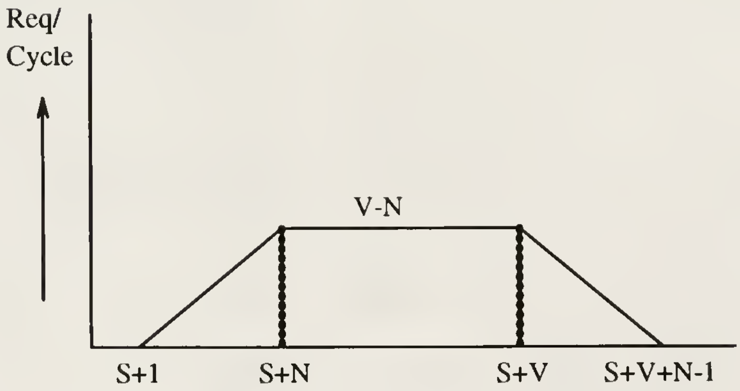

Figure 4.9. Throughput of Dynamic Priority

The first message arrives at memory in a total of $s + 1$ cycles, where s is the number of stages of the network. At each subsequent cycle there is one more processor whose first request arrives. If the vector is at least as long as the number of memory modules, this constant will increment continuously until all processors have one arrival per cycle. It starts decrementing at a constant rate of one processor per cycle once the first processor finishes its vector. If the vector is shorter than the number of memory modules, the curve still has the same shape but some processors will have finished their entire vector even before others have begun. Figure 4.9 depicts the case in which a vector larger than the number of memory modules is accessed in a network of 64 × 64 processors. From this the average vector delay is given by:

$$\begin{aligned} AvVectorDelay &= (\frac{1}{N})(\sum_{i=1}^{N}(s+v+i-1) \\ &= s+v-1+(\frac{N+1}{2}) \end{aligned}$$

where s is the number of stages in the MIN, N is the number of processors in the network, and v is the number of elements in the vector.

From the simulation results it is clear that the CR traffic has benefits from the Dynamic Priority Scheme and suffers a lot under the conventional switch design. This scheme gives continual priority to one port until the requests from that port are exhausted. Thus the utilization of the links can be overlapped and the overall

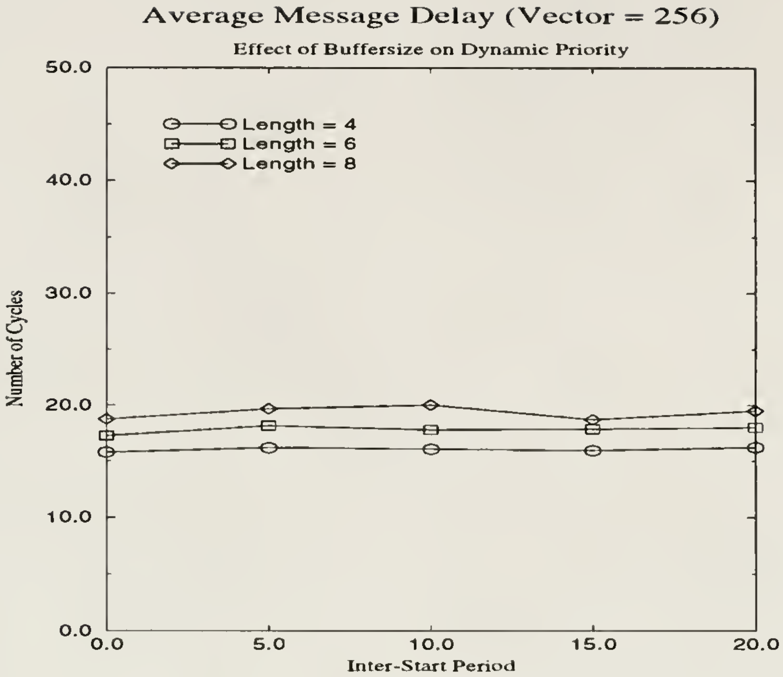

Figure 4.10. Effect of Buffer Size on Message Delay

turnaround time for all the processors is reduced compared to that of the the round robin strategy.

The size of the buffers inside of the switches does not have a great impact on the overall performance of Dynamic Priority scheme. As Figure 4.10 shows, for a vector of length 256 the average message delay degrades slightly as the buffer space increases. The improvement in the average vector delay only becomes significant in the asynchronous case (as can be seen in Figure 4.11). However, even with a small amount of buffer space inside a switch, the proposed schemes perform well.

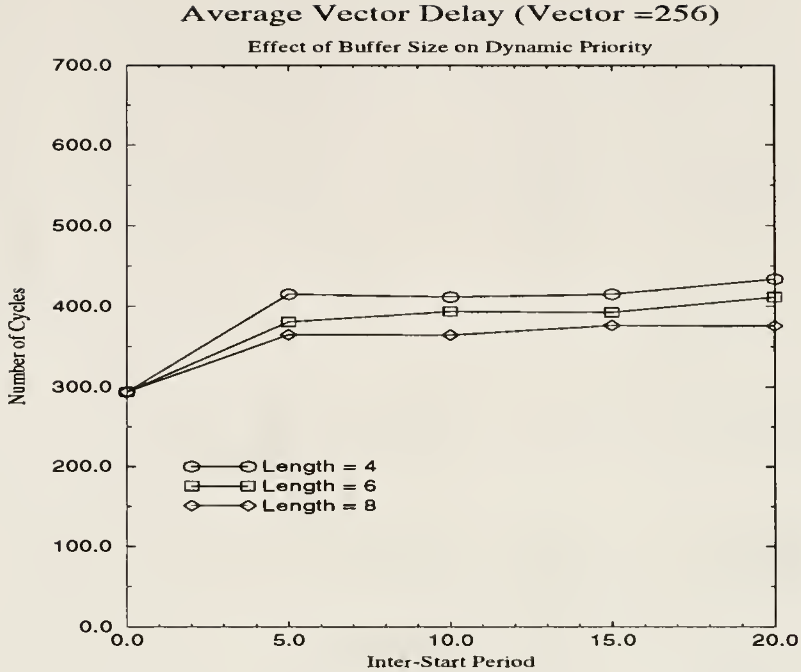

Figure 4.11. Effect of Buffer Size on Vector Delay

4.4.2 Dynamic Priority and Stride Based Mapping Results

In this section, results obtained on CR sequences accessed with strides besides unit stride are discussed. We have limited our experiments to CR sequences with the same bank offset traffic and no load on the network prior to the CR access.

The degradation caused by non-unit stride access, as described in section 4.3, can clearly be seen in Figure 4.12. From the same figure it can be observed that even for the conventional switch design, a skewed storage scheme is able to improve the performance of several non-unit stride accesses. It is not surprising that the disparity between different allocation schemes (interleaved vs. skewed) gets larger as the access stride increases. The main cause for this is that Effect (3) is quite overpowering, and with a larger stride, it also means that fewer modules are accessed in an interleaved memory.

The effect of the stride, and the evaluation of the stride dependent mapping scheme in an interleaved memory can be seen in Figure 4.13. By merely adapting the stride dependent mapping, an adequate improvement over the original bit reverse mapping in an interleaved memory can be obtained. An important observation to be made from these results is that even though the number of modules accessed remains the same (so Effect (3) is not changed), the distribution of the requests itself can cause the improvement. When Bit-Reverse mapping is performed without taking the stride into account (i.e. by default unit stride), all the mapped addresses will be in one corner of the address spectrum, causing a localized multiple hot spot area. The stride dependent mapping distributes these highly requested addresses, and by doing so the requests are also distributed in these directions causing less contention for similar MIN links.

In the event that no skewed storage is available, the best route to take is still to employ the Dynamic Priority. This can be concluded from Figure 4.14, where for a stride 4 access the Dynamic Priority scheme is certainly the one which yields the lowest average vector delay.

To be complete, we provide the graph which shows that the uniform traffic has the same message delay in all four schemes. Compare it to the minimum message delay curve which is the theoretical minimum, assuming there are no conflicts inside the network (see Figure 4.15).

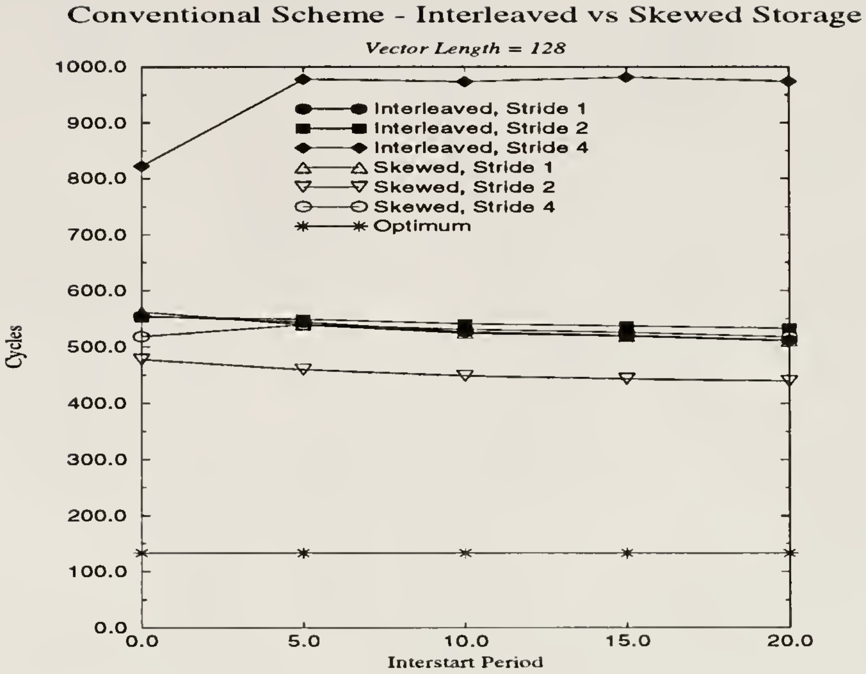

Figure 4.12. Conventional Scheme - Interleaved vs Skewed

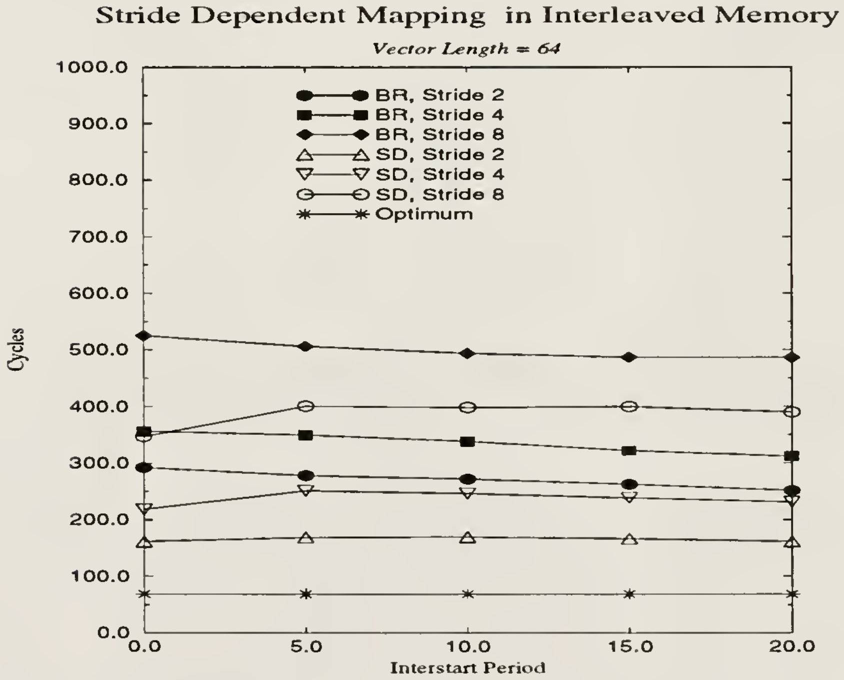

Figure 4.13. Stride Dependent Mapping in Interleaved Memory

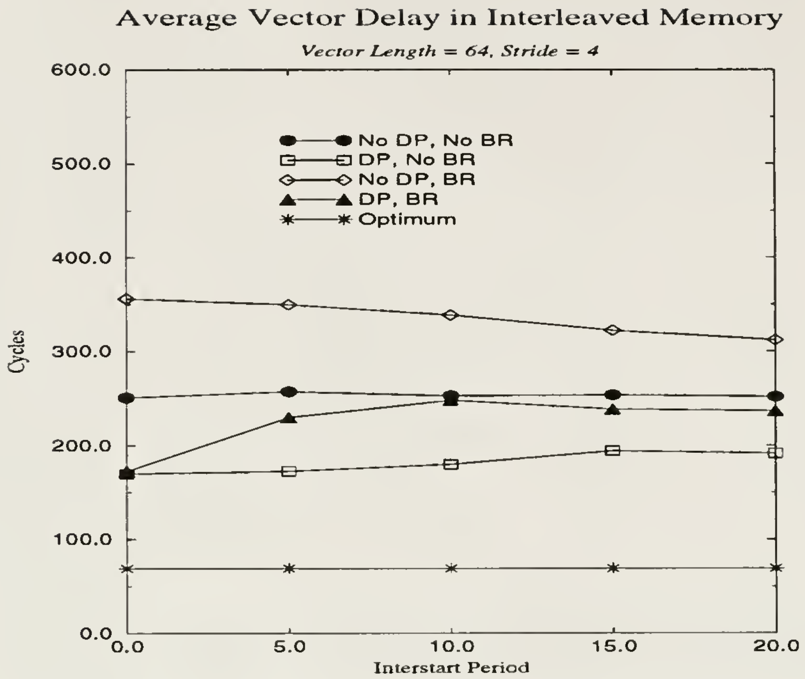

Figure 4.14. Average Vector Delay in Interleaved Memory

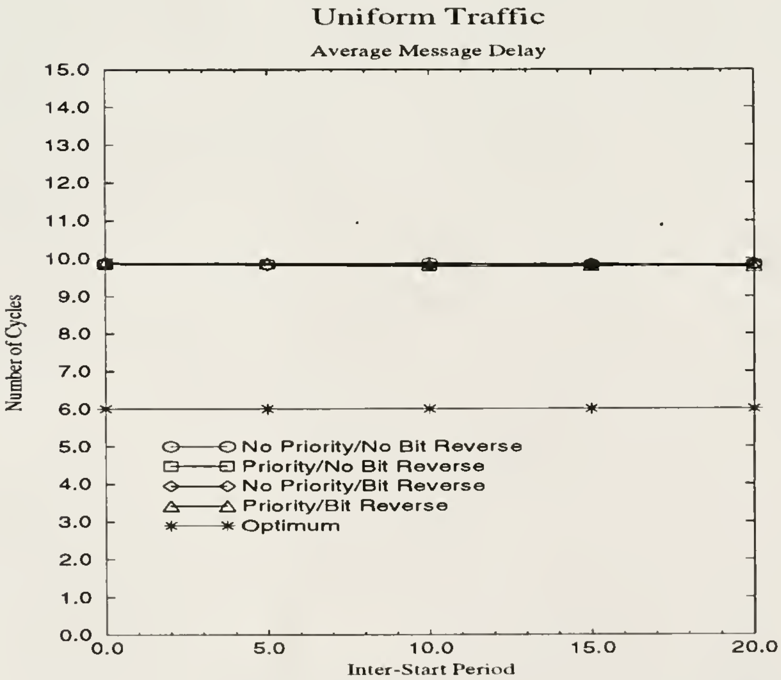

Figure 4.15. Uniform Traffic Under Proposed Schemes

4.5 The Effects of the Reverse Network on CR Traffic

The overall system performance can be measured in terms of the message throughput which can be achieved by the MIN. In practice, the message throughput is not only determined by the bandwidth of the network but also by its latency. A processor may not be able to place a new request until it has received the response to a previous one. In the case of CR accesses one might consider three cases upon which a new CR sequence can be issued:

- As soon as the entire previous CR sequence has been issued, with no need to wait for its return.
- As soon as the first message of the previously requested CR or the element which caused the processor to issue the sequence (for example in the case of a cache miss) sequence has returned.
- As soon as the entire previous CR sequence has returned from memory.

These cases have an impact on the utilization (the processor might not be able to do anything useful until the requested message has returned).

4.5.1 Processor Model

For the purpose of the discussion in this section, we assumed the following processor model:

- All processors have the same workload, and will thus request on the average the same number of CR sequences.

- The CR sequences will be accessed with the same stride, namely 1.

4.5.2 Memory Model

We assume the memory to be interleaved and that all requests to memory have identical latency (we do not distinguish between reads and writes), other than of course the individual latencies incurred due to the network conflicts. In reality, these requests result in distinct traffic patterns on the forward and reverse networks. Reads generate a request of one word which traverses the forward network but returns with two words over the reverse network. Writes on the other hand generate two words over the forward network and a single word travels back over the reverse network to convey the success of the write to the processor. Though this asymetric behavior may cause a difference in the performance in the types of memory traffic, we will not consider this in our experiments.

Each memory module consists of four memory banks, but only one input buffer and one output buffer. After being serviced in a memory bank, messages contend for the reverse network and are queued according to their finishing times in the memory module (FIFO).

4.5.3 Network Model

It is of importance to study the behavior of the CR traffic in the forward as well as the reverse network, and to observe what improvements the proposed schemes can offer. All the permutations of the schemes are worthwhile of study, except one. It is not meaningful to implement the Bit Reverse mapping in the backward network; hence, we restrict this mapping only to the forward network.

We assumed the forward network to be an Omega network and varied the reverse network from an Omega network to a Baseline network. The consequences of using a different type of MIN, in particular as the backward network is investigated and found to be of interest. The type of reverse network and the schemes used in both the forward and reverse network have a direct bearing on the performance obtained.

4.5.4 Performance Evaluation

The delay incurred per CR sequence is a direct measure of the processor utilization, since it is generally assumed that the processor idles as long as there are messages pending. In our experiments, the average delay incurred in the forward network, and the average delay incurred in the reverse network were measured separately for comparison. Table 4.2 lists the parameters used in our model and table 4.3 lists the measures obtained.

Parameter	*Value*
Number of CR Sequences per Processor	1
Memory Latency	4
Number of Processors	64
Number of Memory Modules	64
Number of Banks per Memory Module	4
Buffer Size per Switch	4

Table 4.2. Simulation Parameters

Performance Measure
Average Forward Vector Delay
Average Reverse Vector Delay

Table 4.3. Performance Measures

Here, the average forward vector delay is taken to be the same as defined in Section 4.4. The average reverse vector delay however, includes the memory latency. The sum of the forward and reverse vector delay results in the total average vector turn around time. In this section, we will present the results obtained from the forward and the backward network in a segregated fashion in order to illustrate the usage of different backward network topologies.

In the following, the notation $x/x/x$, is used to indicate the implementation of Bit Reverse in the forward network, Dynamic Priority in the forward network, and Dynamic Priority in the reverse network, respectively. The values for x are E for *Enabled*, or D for *Disabled*. For example, $E/D/E$ refers to the case where the forward network has the Bit-Reverse mapping implemented and the Dynamic Priority scheme disabled. The reverse network, on the other hand, has the Dynamic Priority enabled.

The performance of Dynamic Priority in the forward network depends on what type of backward network is used (compare Figures 4.16 and 4.18). In the first one, Dynamic Priority schemes outperform all the other six combinations whereas in the latter one the Hybrid schemes and the Dynamic Priority schemes seem to have similar results. From these two figures it is also apparant that the Bit Reverse schemes have different outcomes depending on the type of reverse network used. We shall return to this later.

Implementing Dynamic Priority in the reverse network can be beneficial as well although the nature of the requests depends on what schemes were used in the forward network. Dynamic Priority allows the quick passage of requests headed in the same general direction and sequences with long DIRs benefit most from this scheme. The nature of the returning traffic which is generated at the memory modules and which is heading towards the processors is dependent on the order in which they arrived at the modules.

The choice of multistage interconnection network together with whether the Bit Reverse mapping was used or not, have a big impact on the latency suffered by the CR traffic. Note that the schemes which implement Bit-Reverse in the forward network perform worse than the conventional scheme when the Omega network is used, instead of the Baseline network (contrast Figures 4.16 with 4.18, and 4.17 with 4.19). The reason for this phenomenon lies in the mapping performed and the topology of the reverse network. The observations can be explained by means of Figure 2.1. When Bit Reverse mapping is used in the forward network, the modules to which the consecutive requests are mapped, are connected to the same switch in the backward network (if an Omega Network is used as the backward network). Since the contending messages are headed for the same processor, they will be contending on the whole way back. This same phenomenon will occur for all pairs of messages.

Any different backward network which does not connect the modules described above in the first stage would remove this phenomenon, and the Baseline network is an example of a network which meets this criterion.

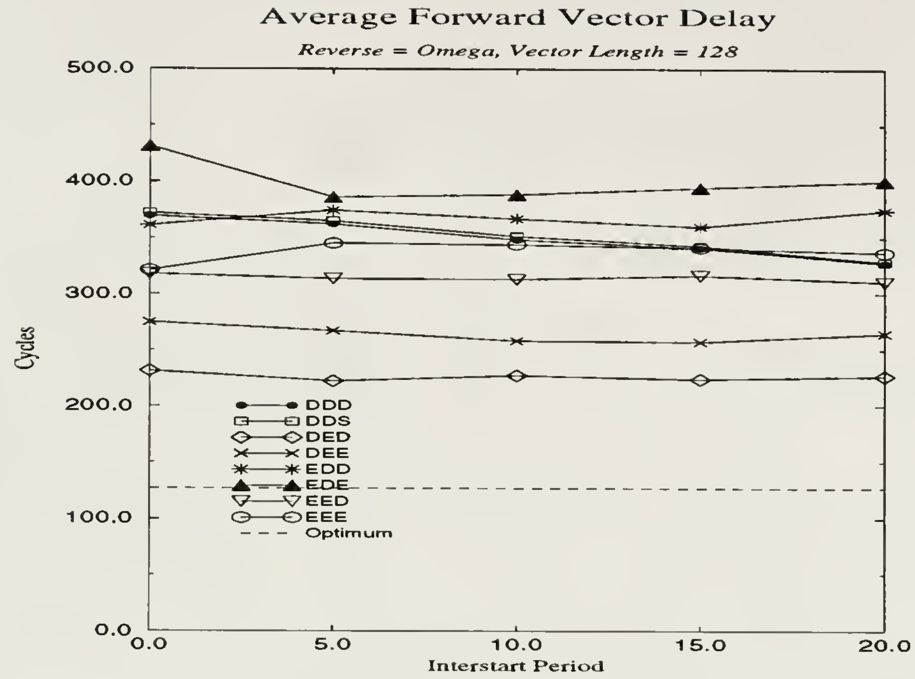

Figure 4.16. Average Forward Vector Delay - Omega Reverse Network

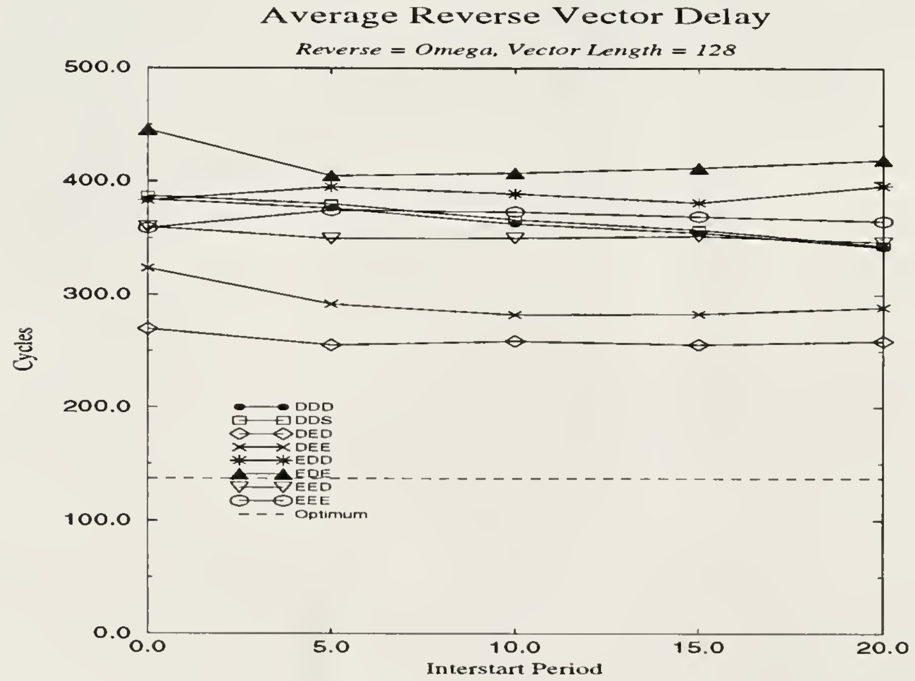

Figure 4.17. Average Reverse Vector Delay - Omega Reverse Network

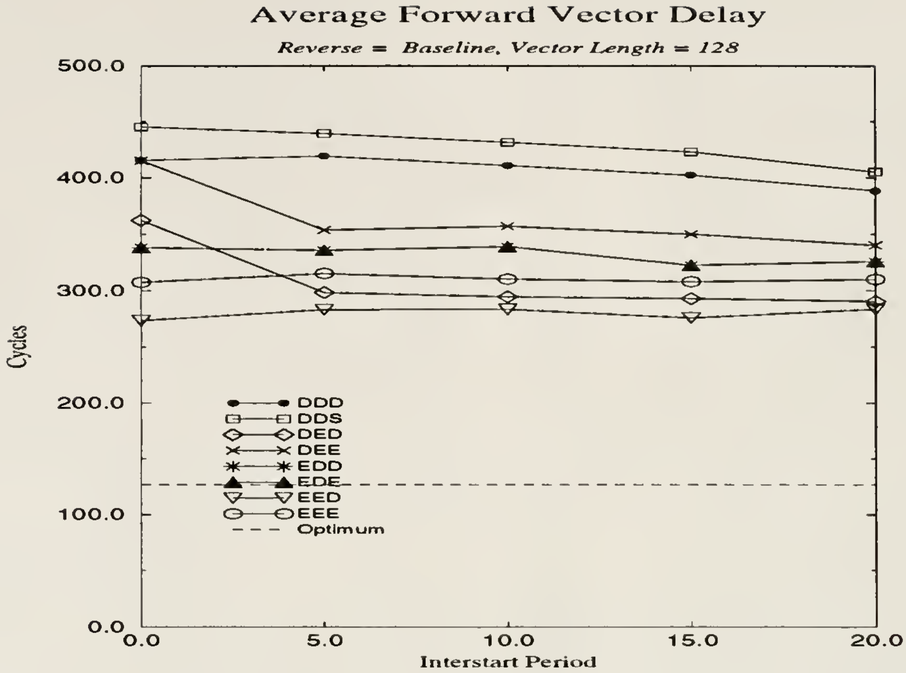

Figure 4.18. Average Forward Vector Delay - Baseline Reverse Network

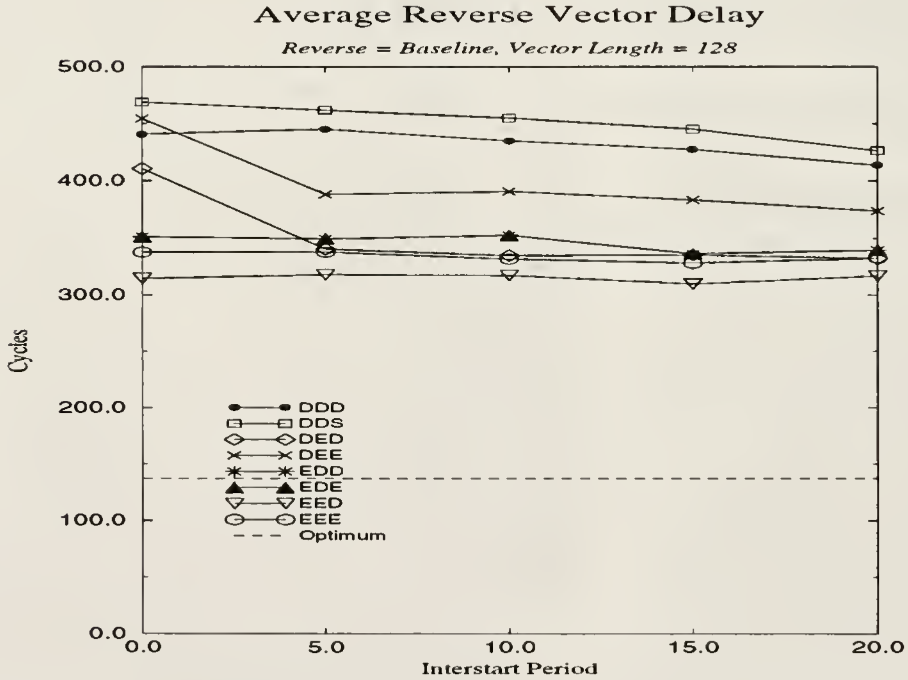

Figure 4.19. Average Reverse Vector Delay - Baseline Reverse Network

4.6 Chapter Summary

In this Chapter, the Consecutive Requests traffic pattern was discussed and the causes for network deterioration under CR traffic have been determined. Two methods were discussed to alleviate the poor performance of CR traffic in a MIN. The first method is a scheduling mechanism that would replace the round robin scheduling, namely the Dynamic Priority. The second method, Bit-Reverse mapping, is a mapping scheme with the purpose of moving the link contention to the latter stages of the network. It is shown that these schemes can improve the performance of the network under CR traffic considerably. In the following Chapter, the issues involving the interaction of CR traffic with regular accesses are addressed.

CHAPTER 5
VECTOR/SCALAR INTERACTION IN MINS

In Chapter 4 the problem of access conflicts due to similar CR pattern accesses was studied. Though this problem in itself deserved the attention it received, a more common processing model is one in which processors access different types of data, resulting in a mix of interacting traffic patterns. Memory traffic generated by processors can be characterized by bursts of requests which may belong to either a vector or a scalar stream. The network performance in this case, will depend on the rate of issue of each of these request streams, the type of access (traffic pattern) made by each of the processors, as well as the degree of interaction of these multiple streams.

The objective is still that the requests issued are to be satisfied promptly by memory. From the previous chapter it can be gathered that when left alone, CR traffic can be detrimental in MINs. The question now remains whether the performance of this type of traffic is made worse by the presence of non-CR traffic patterns. Some of these non-CR traffic types are for example, uniform traffic, and hot spot traffic. The reverse is absolutely of interest: whether the presence of CR traffic has a degrading effect on non-CR traffic. Non-CR traffic will be referred to as scalar traffic in the remainder of this chapter.

One example in which vector accesses are interspersed with scalar references can be drawn from a parallel algorithm also known as the Gaxpy algorithm. In the shared memory Gaxpy computation, a vector z is computed as the sum of a vector y and the product of a matrix A and vector x. A, x, and y reside in shared memory. We assume throughout that the shared memory multiprocessor has n processors. The following example is that of the n-by-n *gaxpy* problem $z = Ax + y$ partitioning.

$$\begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} A_1 \\ \vdots \\ A_n \end{bmatrix} x + \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

For simplicity we assume that there are n processors and that each processor is allocated one element from each vector and one row of the matrix A. From the algorithm in Figure 5.1 it can be observed that each processor engages in the fetch of a scalar element y_i and two vectors x and A_i. The vector x is a shared read only vector which all processors will load into their local memory. The vector A_i, which pertains to a row of the matrix A, is not shared but different elements of the different rows may still reside in similar memory modules, causing potential access conflicts. (We assume that the matrix is stored in row major). Note that after each processor completes the computation, it returns the value to the shared memory by a *store* statement.

Algorithm 5.1 Gaxpy

```
begin
    Shared Variables [A(1:n,1:n), x(1:n), y(1:n)]
    Local Variables $x_{local}(1:n), z_{local}, a_{local}$
    for id=i,n do in parallel
        load x into $x_{local}$ //vector load //
        load y(id) into $z_{local}$ //scalar load//
        for j=1,n
            load A(id, j) into $a_{local}$
            $z_{local} = z_{local} + a_{local} \times x_{local}(j)$
        end-for
    end-for
end gaxpy;
```

The remainder of this chapter is organized as follows. Section 5.1 describes the nature of the vector-scalar interaction. Section 5.2 depicts the model used to evaluate the performance of the vector-scalar interaction, and Section 5.3 details the performance results obtained by implementing the Dynamic Priority and Bit Reverse mapping schemes. Section 5.4 summarizes the findings in this chapter.

5.1 The Effects of Vector/Scalar Interaction

Both vector and scalar request streams can be adversely affected by one another when they interfere with each other for a long period of time. The switching elements of a MIN may experience an unbalanced traffic and in order to understand the nature of this interference we will consider the network behavior which is caused by elements of scalar streams and CR (vector) streams. This interference has the following characteristics:

1. A sequence of uniform and independent requests (scalars) which are generated at a source node interact with a sequence of requests destined to consecutive sink nodes. We will assume that the scalar references are generated in an independent and uniform fashion.

2. Consecutive requests are generated in a continuous fashion. Scalars, on the other hand, have a probability of being generated in a given cycle. The probability that a scalar reference will be made is also referred to as the scalar issue rate.

The effects and performance of uniform traffic in MINs have been studied in literature. The detrimental effects caused by vector/vector interaction was the subject of the previous chapter for which the Dynamic Priority and Bit-Reverse were proposed. The first one is an alterative to the round robin scheduling strategy and the Bit Reverse mapping which reverses the addresses for unit stride accesses.

When vectors and scalars interact, it would seem likely that the scalar requests would be penalized the most. Uniform scalar accesses will find that the paths which they must take to be mostly saturated with requests from vector streams. It is empirically found that when the percentage of processors actively accessing vectors is equal to the percentage of processors engaged in scalar requests, the volume of requests generated by vector accesses is sufficient to congest the paths which are shared by the scalar requests.

Insertion conflicts in individual switches of the MIN can be handled in various ways. When the round robin strategy is used, vector requests and scalar requests will be interleaved and the consecutive nature of the vector requests will be altered. With larger networks this scalar interaction will cause vector sequences to be broken up and the DIRs to be shortened. The interaction of vector sequences (nonuniform traffic pattern) interspersed with uniform scalar references results in a more or less uniform

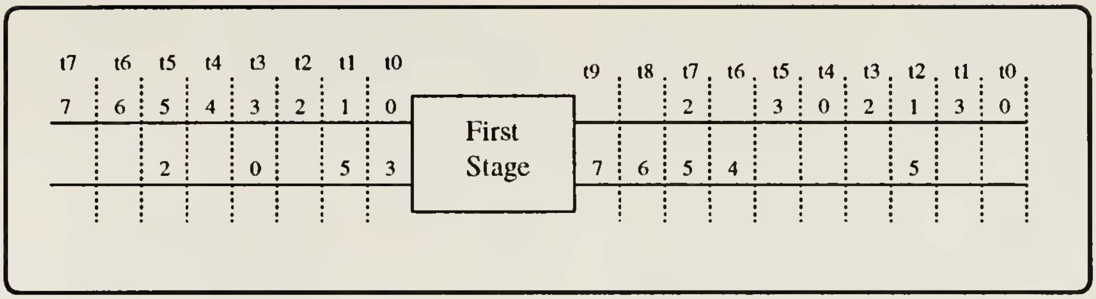

Figure 5.1. Vector-Scalar Interaction Example

traffic pattern and eliminates the congestion normally accompanied by nonuniform traffic patterns. It can hence be expected that the interaction of scalars with vectors can improve the performance of the vector access, at the cost of the scalar references.

Consider for example the activities in a switch in the first stage when a CR sequence arrives at the upper input port and a scalar sequence arrives at the lower input port, with an arrival rate of 0.5 (one arrival every two cycles). For example, in Figure 5.1, the upper input port receives the CR sequence 000, 001, 010, ... , 111 (8 × 8 system) in a continuous fashion while at the lower input port there is an arrival of a random sequence of destinations. The result of a round robin strategy in this switch results in sequences at the output ports which no longer exhibit a long DIR (which, as we conjectured from the previous chapter, is the main cause for the poor performance).

By implementing the Dynamic Priority scheme the issue of fairness becomes immediate. Since Dynamic Priority gives priority to a single port until that port is totally exhausted, the port at which the vector sequences arrive will undoubtedly be served with a higher priority, unless the arrival rate of the scalars is one. In the average case, a scalar element will have to wait for half the vector sequence, if the current priority receiving port happens to be the one at which the vector sequence is arriving. Vector sequences on the other hand, will benefit from the interaction due to the randomization effect as was explained earlier.

The Bit Reverse mapping has a specific purpose which is that of dispersing the consecutive accesses to non-consecutive locations. When this mapping is applied to uniform scalar accesses the result is another uniform set of accesses. The interferences of these two types of requests, even after the mapping, will have no remarkable effect on one another. The scalar requests will be subject to the congestion caused by the volume of vector requests, as was described above, but that is not due to the mapping.

5.2 Vector/Scalar Interaction Models

For the discussion of our processing model we made the following assumptions (see Table 5.1). We are only considering the interaction in the forward network, and thus assume that the memory bank buffers to be infinite. This way, memory can handle all the requests, and there will be no queueing in the forward MIN due to busy memory modules.

The simulation model then proceeds to issue vectors with equal length, unit stride, and with the same bank offset. The interacting processors are continuously either issuing vectors or scalars (following one of the request models described below). Each measured point of the simulation results represents the average of 200 runs, with each run simulating 10000 operating cycles.

The models which we will be using and measuring are:

Simulation Parameters
No scalar caches
Infinite memory bank buffers
Scalar traffic is uniform
Vectors of equal length
Vector stride of one
Same bank offset
Interleaved memory
Forward network only

Table 5.1. Assumptions of the Vector/Scalar Model

- Single Request Mode (SRM)

- Dual Dynamic Request Mode (DDRM)

- Single Static Request Mode (SSRM)

The *Single Request Mode* model is followed by the Cray X-MP system where vector instructions cannot be issued as long as scalar memory accesses are still in progress. The SRM model requires barriers to be placed for all processors participating. It also places the most stringent requirements on the network. In a way, the interaction of scalar and vector accesses is not an issue because they will never interact. What is of essence in this model is the latency incurred during both the scalar burst as well as the vector burst, as this will be the decisive factor. For this reason, we shall not consider this model one in which the desired interaction is manifested.

In the *Dual Dynamic Request Mode* (DDRM) model processors can issue both vector and scalar streams, without having to wait for either to have completed before the other stream can start. This is an idealized situation, assuming the processor has enough buffers to have multiple requests outstanding. The last model, the *Static*

Single Request Mode (SSRM), is one in which the processors are dedicated to one particular request mode, instead of being general purpose. For the duration of the experiment, a processor has a single request mode which remains unchanged.

The request mode affects the issue rate and the dependencies between requests from the same stream. Scalars are issued according to a burst rate and are independent of one another, whereas individual vector requests are dependent and are issued in a continuous fashion (CR traffic pattern). Moreover, the number of processors which are each issuing either scalar or vector traffic also has a direct bearing on the performance.

In the DDRM model, processors can request both vectors as well as scalars, but not simultaneously. A processor can start either a vector or a scalar request. The probability with which a processor will do one or the other should be a parameter to be varied. The duration length of a scalar stream is drawn from a burst interval. The frequency of scalar requests during this scalar burst is decided by the scalar issue rate. Scalar bursts and vector bursts are interleaved, so a scalar burst is always followed by a vector access and a vector access is always followed by a scalar burst. The processing model assumes enough buffers are provided to have all requests outstanding which may still be in the network or are being serviced by memory. The end of a vector access is indicated by the issue of the last vector element, upon which another scalar burst is started. The end of scalar burst is indicated by the system clock, and any of the remaining requests which are still pending in the system are allowed to finish and do not impede the vector access to start. The scalar burst length is a direct factor of the duration of interference, whereas the scalar issue rate directly affects the degree of interference.

In the SSRM model, the request mode of each processor is fixed. In contrast with the DDRM model, a vector requesting processor issues vectors which are not followed by scalar requests but instead are followed by an idle period. Scalar issuing processors issue scalars according to a certain issue rate, and for simplicity we will assume that the issue rate is the same for all scalar requesting processors. The interaction now is defined by the locations and the number of the scalar requesters with respect to the vector requesters. The simplest assumption to make is that the request mode is chosen in an independent and random manner so the locations of these individual processors are random. Together with the scalar issue rate the scalar/vector interaction has tangible parameters which can be varied and observed in the experiments we shall conduct.

5.2.1 Special Interaction Types

In this section we shall describe the traffic patterns which may arise from the DDRM model:

- Scalar burst length of 0. The interaction here degenerates to a vector/vector interaction where there are continuous vector references (there are no scalar references between vectors). The traffic generated resembles that of a single vector access.

- Scalar issue rate of 0. The vector sequences will be interacting with other vector sequences, and vector sequences are followed by a period of no scalar references.

- Scalar issue rate of 1. This results in the most severe interaction between scalar and vector references. The scalar references are issued continuously but in an independent fashion. They will serve primarily as a means to randomize the vector sequence.

Special traffic patterns which arise from the SSRM model are:

- Scalar burst length of 0 or issue rate of 0. The interaction becomes one in which vector/vector interaction takes place but in addition to that, there are idle processors (scalar requesters are idle).
- Scalar rate of 1. The interference caused by a continuous sequence of scalar references and a vector reference is similar to that of the DDRM model with s scalar rate of 1. The vector sequence will be randomized by the interference of the uniform traffic.
- Percentage scalar requesters of 0, this results in a vector/vector interaction model, such as was studied in the previous chapter. The vector sequences will be interleaved by idle periods.

5.3 Performance Evaluation

The simulation model used was a 64 × 64 buffered Omega network. The buffer capacity of the switches was held at 4 messages per port. We assume that a processor can continue issuing messages as long as the input register to the first stage is available. When a CR sequence is being issued, messages are generated every cycle, but scalar requests are generated according to a scalar arrival rate. The programming model which we are investigating assumes that the processors in the system are executing parallel tasks which have been previously allocated. Processors execute and request scalars and vectors in an interleaved fashion. We assume that the vectors are of the same length and are accessed with the same stride. The scalar access burst length is selected from an uniform distribution. The time at which the processors start their access (vector or scalar) is also varied, from synchronous to asynchronous. We have limited the experiment to vector references which start with the same bank offset.

The model primarily performed comparison experiments between the conventional switch and the Dynamic Priority and Bit Reverse mapping schemes. The measures which were of interest were the average scalar throughput and delay, and the average vector throughput and delay. The scalar (vector) throughput is defined to be the number of scalar (vector) messages which arrived at memory per cycle. This chapter will concern itself with the forward network only, and hence the memory is assumed to be able to service all the requests. The average vector delay is defined to be the average number of cycles it takes for an entire vector to reach memory. The results obtained are then compared against the optimum, where the optimum is obtained by assuming that there are no conflicts in the network.

5.3.1 Performance of SSRM Model

We assume that per experiment, the number of processors which request vectors is fixed. We also assume that the idle interval, vector length and stride are identical and

Parameter	*Value*
System Size	64 × 64
Vector Length	64
Buffer Size per Switch	4
Access Stride	1
Scalar Issue Rate	0.50
Idle Interval	100
Ratio Scalar/Vector Processors	1:1

Table 5.2. SSRM Model Parameters

fixed amongst all vector requesting processors. Unless explicitly given, the parameters used in the SSRM experiments are assumed to be the ones given in Table 5.2.

Giving scalar references a higher priority will result in a degradation of vector performance. It can be observed from Figure 5.2 where four schemes used scalar priority and as the scalar issue rate increases, the delay incurred by vector sequences gets worse. The curves in Figure 5.3 reflect this same information, but shows the scalar delay for the same schemes as in Figure 5.2. As expected, the highest scalar delays are incurred for the scheme which did not give priority to scalar references.

The number of processors actively engaged in vector requests decides the degree of contention in the system. By giving scalar references a higher priority, all the four schemes have more or less the same performance (see Figure 5.4), except when the vector load is low. In that case, the conflicts in the early stages can be removed effectively by using the Bit Reverse mapping. Scalar requests can randomize the sequence in the latter stages and together provide an excellent turnaround time. In Figure 5.5 the scalar references yield similar results, except that in the conventional scheme the delay is slightly higher for a large percentage of vector requesting processors.

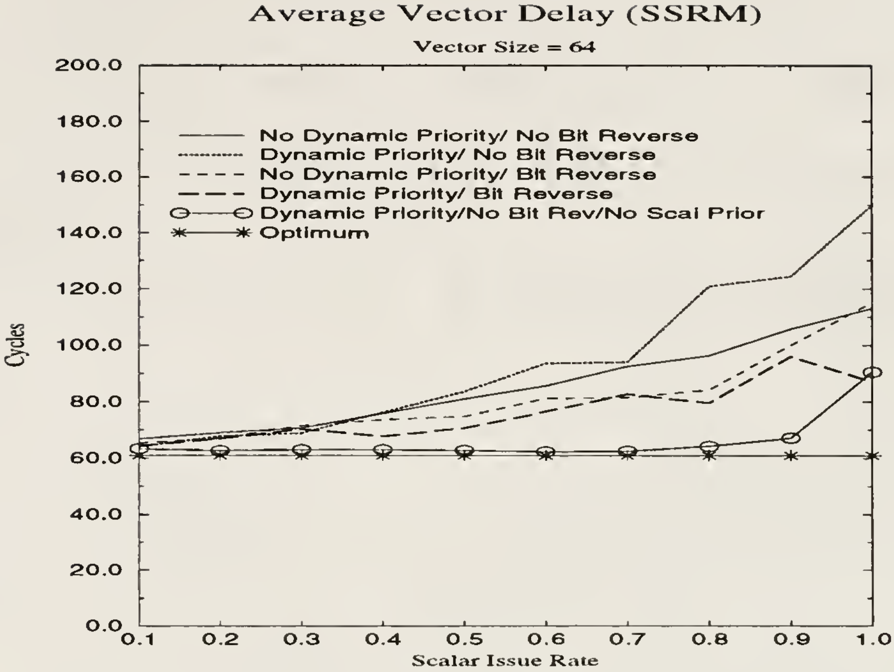

Figure 5.2. Average Vector Delay - SSRM Model

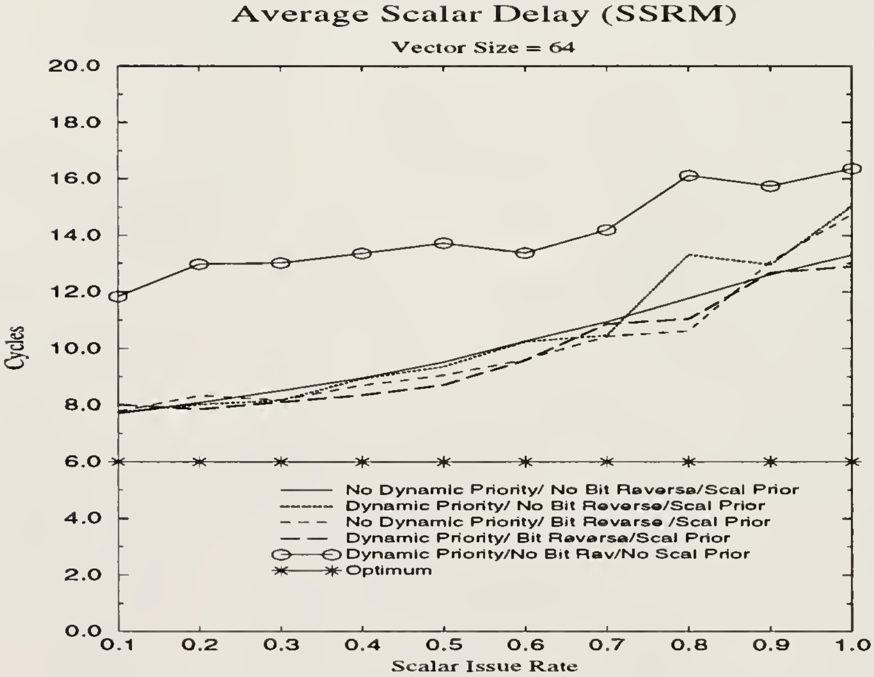

Figure 5.3. Average Scalar Delay - SSRM Model

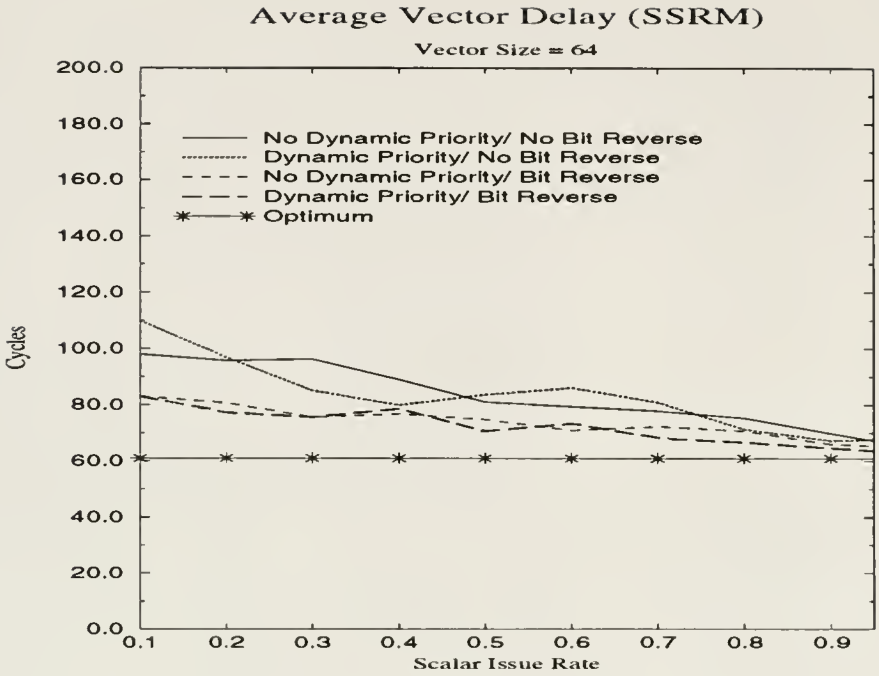

Figure 5.4. Average Vector Delay - SSRM Model (Scalar Priority)

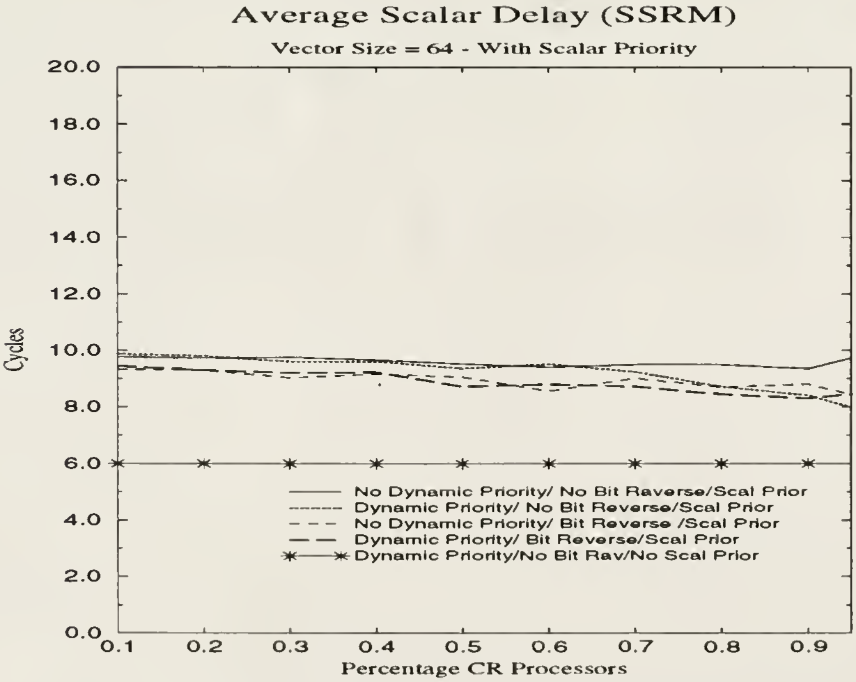

Figure 5.5. Average Scalar Delay - SSRM Model (Scalar Priority)

Parameter	*Value*
System Size	64 × 64
Vector Length	64
Buffer Size per Switch	4
Access Stride	1
Scalar Issue Rate	0.50
Scalar Burst Length	0-100

Table 5.3. DDRM Model Parameter

5.3.2 Performance of DDRM Model

The assumptions made are that each processor had an equal probability of starting either a vector or a scalar stream. Ideally, this will depend on the application, since most vectorized code does not contain many of scalar accesses, perhaps the proportion of scalars with respect to vectors is slightly exaggerated. All vectors were assumed to be of the same length and accessed with the same stride. We assume that the vectors are being accessed from from the same bank and that the scalar requests are uniformly distributed over the entire memory spectrum.

The length of the scalar burst is a decisive factor on the number of vector accesses in the model, since the vector accesses are interleaved with scalar bursts. Unless otherwise specified, the assumptions made in the DDRM experiments are the ones listed in Table 5.3.

The results in Figures 5.6 and 5.7 were obtained from issuing vectors of length 64 with a scalar request rate of 0.5. The scalar burst length was uniformly drawn from an interval of 0-100 cycles. In this experiment, vector requests were not distinguished from scalar requests, which made the scalar delay considerably higher in the case

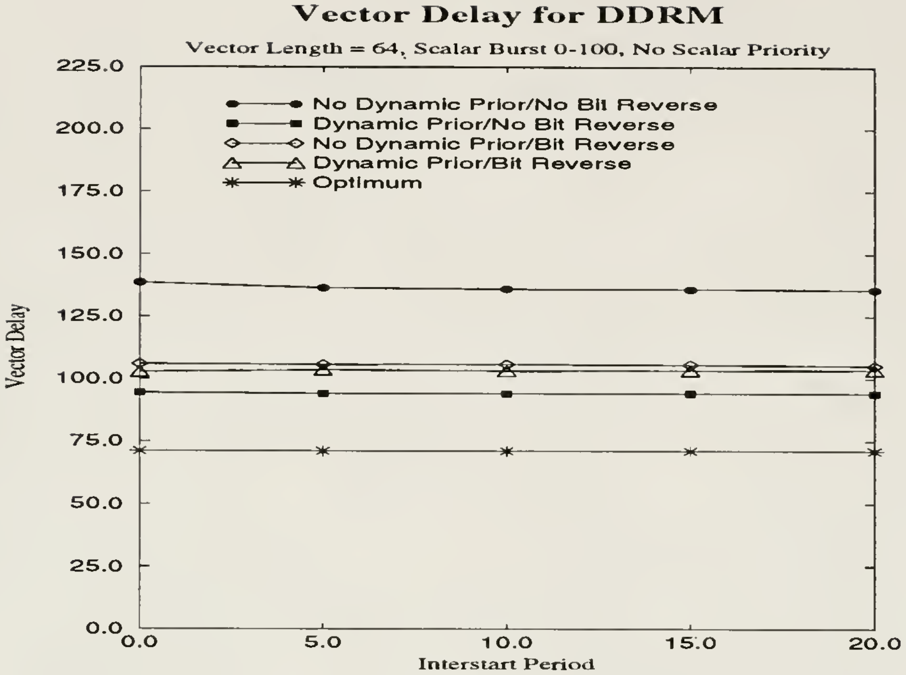

Figure 5.6. Average Vector Performance for All Schemes

of Dynamic Priority (Figure 5.7). Vector accesses however, were quite improved compared to the conventional scheme. Figure 5.6 confirms the fact that even with scalar interference, both the Dynamic Priority and the Bit Reverse schemes can outperform the conventional scheme, as was found in the previous chapter.

The scalar burst parameter not only has direct bearing on the vector access, it also allows the study of special cases such as the scalar burst length equal to zero (hence the study of continuous vector accesses) and scalar issue rate equal to one (which means the issue rate of scalars and vectors are the same). When the scalar issue rate is one, it can be seen as studying the temporal correlation of CR patterns, since for a period of time there will be spurious accesses which do not correlate identically with the ongoing vector access.

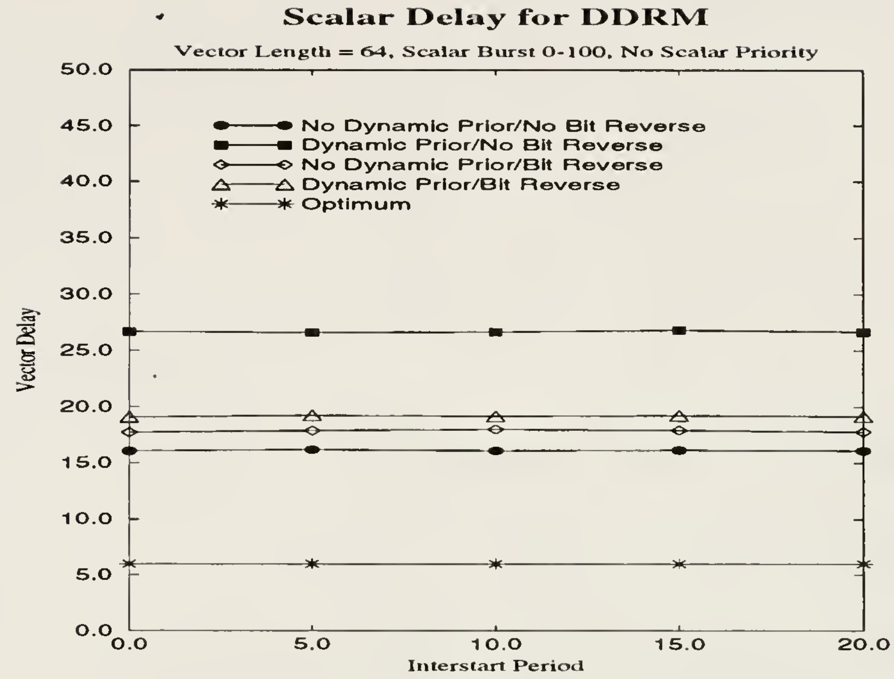

Figure 5.7. Average Scalar Performance for All Schemes

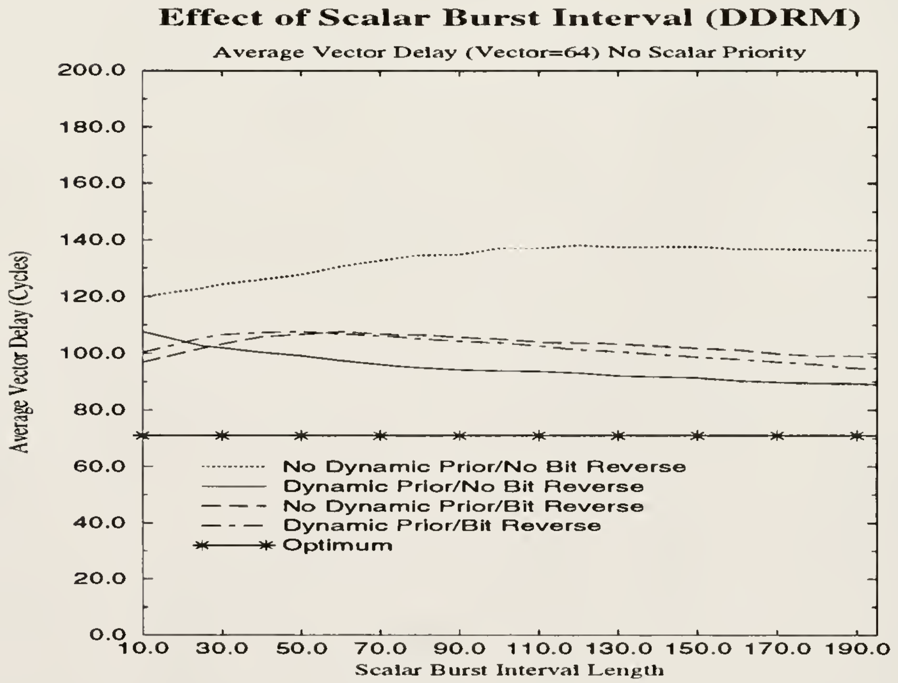

Figure 5.8. Effect of Scalar Burst Length on Vector Delay - No Scalar Priority

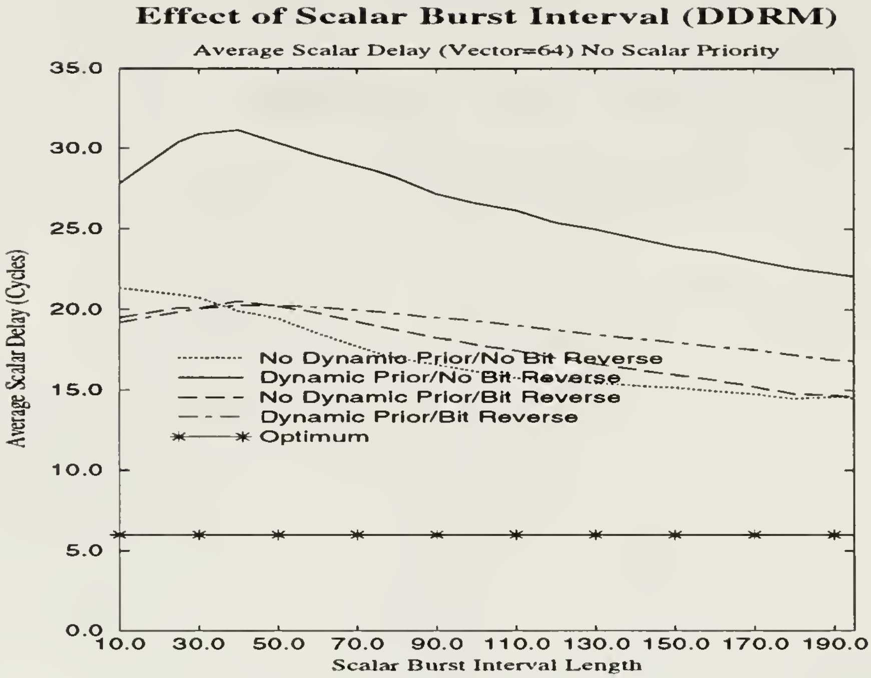

Figure 5.9. Effect of Scalar Burst Length on Scalar Delay - No Scalar Priority

In Figure 5.8 the average vector delay for the conventional scheme becomes increasingly worse even under long scalar burst intervals. Long burst intervals reflect a large number of scalar requests, and a long vector/scalar interaction period. The Dynamic Priority, Bit Reverse and the Hybrid schemes can offer a significant improvement on the vector performance. As was expected the Dynamic Priority scheme favors long DIRs, which is not advantageous to the scalar requests, as can be perceived in Figure 5.9. The Bit Reverse and Hybrid schemes can offer similar results as the conventional scheme due to the distributing effect of the Bit Reverse mapping.

The scalar issue rate depicts the volume of scalars injected into the system, escalating the scalar/vector interaction. It is hence expected that the average vector delay increases as the scalar issue rate approaches one (Figure 5.10) When the issue

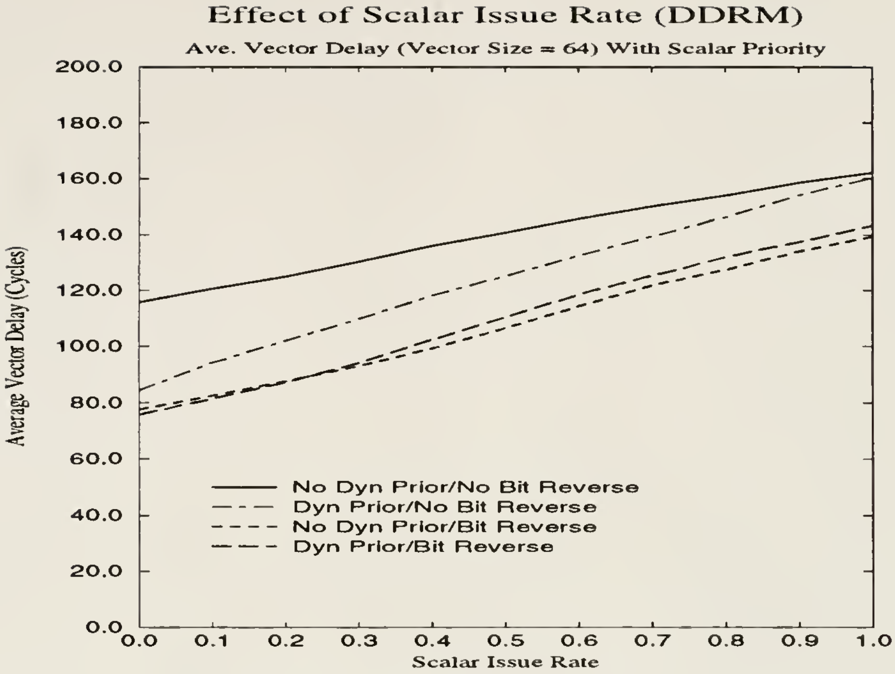

Figure 5.10. Effect of Scalar Issue Rate on Average Vector Delay

rate is zero, the equivalent of having only vector interaction amongst the processors with idle periods between consecutive vector sequences, it can be seen in Figure 5.10 that the conventional scheme is significantly worse than any of the proposed ones. The scalar delay is slightly better in the proposed schemes as long as the issue rate is less than 0.5 but with higher rates, the Bit Reverse and Hybrid schemes perform slightly worse than the conventional one. Due to the scalar priority, the effect of higher scalar issue rate does not affect the average scalar delay dramatically.

5.4 Chapter Summary

The nature of the interaction between Consecutive Requests and scalar references was discussed. Scalar requests break up the long DIRs of CR sequences and thereby causing more overlap of link utilization. Unless priority is given to scalar references, the scalar delay increases due to the volume of CR requests.

CHAPTER 6
SETUP AUGMENTATION PROCEDURES

In nonbuffered selfrouting switches it is a common procedure to set up a connection prior to routing a packet in order to avoid retransmission of lost packets due to the nonbuffering capability. During this setup procedure the two main goals are to setup as many connections as possible (to ensure a high throughput) but at the same time do this with the least possible overhead. A subgoal in synchronous transmission is also to combine path setup procedures with synchronization in order to notify the sources that the entire setup procedure has terminated and that the actual data transmission may begin. The purpose of this Chapter is to improve the throughput of nonbuffered banyan networks.

Related work in this area has usually made the fundamental assumption that at the beginning of a transmission cycle, all setup requests are aligned and processed in a parallel fashion. Link conflicts (inside a switch) are usually resolved in a random manner whereby the victimized requests are dropped and denied a setup during the data transmission interval. Victors continue on to subsequent stages where again they might have output link conflicts. Ultimately, the victors of all stages are granted a connection. This is also known as parallel path setup (PPS), and notably Patel [PATE81] gives the analysis under these assumptions.

Recently a study conducted by Lea [LEA92] showed that an incremental path setup (IPS) procedure can be implemented which can increase the number of possible connections at the expense of having more setup intervals (assuming the parallel path setup scheme only uses a single interval). Lea [LEA92] is able to achieve a spectrum of improvements by increasing the number of setup intervals. In order to achieve a reasonable difference in throughput the number of setup intervals equals the number of contending sources (also called pure IPS). The IPS scheme then attempts to set up each of the sources, one at a time. The order in which the setups occur (setup sequence) places a certain priority on the traffic at these inlets, the highest priority is a priori assigned to the inlet which is to be attempted first and the inlet at the end of the sequence has the lowest priority. Another drawback to the IPS scheme is that the data transmission cannot start until all the setup intervals are over, even though the network might be idle during some of these intervals due to a lack of requests (low input rate).

This has led us to develop a setup scheme whose overhead has the same order of complexity as the PPS overhead but with the same or higher throughput than that obtained by Lea's IPS scheme. This chapter discusses a new parallel path setup scheme. The basic version of this algorithm is equivalent to the IPS system. An improvement to this algorithm has been implemented which yields a higher throughput than the basic version. We present the version which has an embedded synchronization capability. The attractiveness of this scheme is its simplicity, reduction of overhead, and embedded synchronization.

The remainder of this chapter is organized as follows. Section 6.1 gives an algorithm for finding the maximum connections setup for a given input pattern. Section 6.2 outlines the Re-Attempt parallel path setup scheme and its operation. Section 6.3 describes the synchronization method which can be embedded into the ReAPPS scheme. In Section 6.4 the overhead of the ReAPPS scheme is obtained analytically and in Section 6.5 the performance of the ReAPPS scheme is evaluated by means of simulation and contrasted to the IPS and Patel's scheme. In Section 6.6 a summary of the entire chapter is given.

6.1 Optimal Path Setup

Given a specific input pattern (where an input pattern is a list of source and destination pairs), consisting of a set of (s, d) pairs, where s is the source and d is the destination, it is possible to obtain an optimal set of connections which can be established if that input pattern were to set up paths at the same time in a banyan network.

Before discussing the problem of obtaining the maximum throughput in a given input pattern, we note that it is frequently necessary to determine the largest amount of material that can be shipped from a given source to a given destination given a particular network (here, a *network* is a class of directed graphs which will be described in the section below). The next section details the different types of network flow problems that arise in transportation networks.

6.1.1 Transportation Network Flow Problems

Network analysis is a powerful method for accurately representing many real-world problems in many areas. A *network* is said to consist of a set of nodes and a set of arcs (edges) connecting the nodes. Nodes can be classified as *source, transshipment,* or *sink* nodes. Commodities are supplied by sources, and are demanded by sink nodes. Transshipment nodes are nodes which can be both source as well as sink (and are often used as intermediate nodes in the transportation network). Arcs can have a non-negative number associated with them indicating the capacity of the edge.

One of the most important results in the theory of network flows is the *maximum flow/minimum cut* theorem, proved by Ford and Fulkerson [FORD62]. This theorem states that for any single commodity flow network, the maximum feasible flow is equal to the capacity of the minimum cut of the network. The cuts of interest here are defined as set of arcs whose removal will disconnect the source from the sink. The capacity of the cut is the sum of the flow capacities of the arcs of the cut.

Single commodity flows involve units of flow which are indistinguishable from one another. The units are assumed to be supplied by a single source and directed to a single sink. However, if the units need to be distinguished, the problem becomes that of a *multicommodity flow.* An example of a multicommodity flow problem is that of the transportation of products from several sites to several destinations. If the products can be shipped from any site to satisfy the demands of any destination, then the problem can be formulated as a single commodity problem. However, if the shipment is constrained by certain sites-destinations combinations, then the problem is no longer of a single commodity nature.

In general, a multicommodity flow problem can be stated in terms of a network $G(V, E)$, with K pairs of vertices (s^k, t^k), $k = 1, \cdots, K$ are specified as source-sink pairs associated with K commodities.

Let an edge from node i to node j be denoted as (i, j). For each k let

- f_{ij}^k = flow of commodity k on edge (i, j)
- F_k = value of flow to be sent from s^k to t^k
- $f_{ij} = \sum_{k=1}^{K} f_{ij}^k$ = total flow on arc (i, j)

For each node $i \in V$ let

- $\alpha(i) = \{\ j \in V | (i, j) \in E\ \}$
- $\beta(i) = \{\ j \in V | (j, i) \in E\ \}$

A multicommodity flow is a set of flows $\{\ f_{ij}^k | (i, j) \in E, k = 1, \cdots, K\ \}$ satisfying the flow conservation constraints for all k below:

$$\sum_{j \in \alpha(i)} f_{ij}^k - \sum_{j \in \beta(i)} f_{ij}^k = \begin{cases} F^k & \text{if i} = s^k \\ -F^k & \text{if i} = t^k \\ 0 & \text{otherwise} \end{cases}$$

A *capacity* function assigns a non-negative value $c(i, j)$ to each edge (i, j). The flow is assumed to be non-negative as well, i.e. $f_{ij}^k \geq 0$ for all edges (i, j). The capacity constraints state that for each edge $(i, j) \in E$: $f_{ij} \leq c(i, j)$.

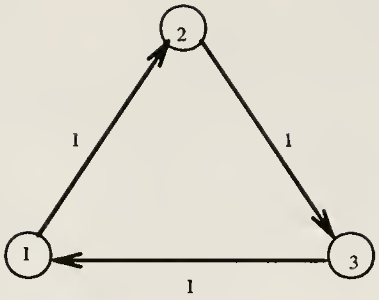

Figure 6.1. A Three Commodity Flow Problem

Multicommodity problems are much more complicated to solve than single commodity problems. One of the main difficulties is that the solutions are not necessarily integral. The classic example in which this is illustrated is given in Figure 6.1. Commodity 1, 2, and 3 originate at nodes 1, 2, and 3 and are destined to nodes 3, 1, and 2 respectively. The capacity of each edge is 1, i.e. the sum of all commodities flowing through each edge cannot exceed 1. The obvious answer to this flow problem is that the maximal flow is 1, since 1 unit of flow of any single commodity blocks the flow of any of the other two. However, the maximal flow is 1.5 units, 0.5 units of each product. The total flow on each edge would still be 1. Thus, the multicommodity flow maximal flow problem does not necessarily have integer valued solutions.

A mulitcommodity flow network must satisfy the *cut condition* in order for the existence of a k-commodity flow to satisfy the demands, but it is not always sufficient. The cut condition states that the capacity of a cut X must be able to meet the demands of the cut ($c(X) \geq d(X)$).

Contrary to single commodity network flow problems, multicommodity flow problems are in general NP-complete for general graphs and integral flow [KARP75]. Polynomial algorithms can be found for certain graph topologies and with real-valued flows, or on a restricted number of commodities [KENN78, EVAN78, MATS85]. No true polynomial time algorithm is known for the multicommodity flow problem on general graphs. Like all network flow problems, the multicommodity flow problem can be formulated as a linear program. Though the simplex method is a practical solution, for some specific types of graphs, special purpose algorithms have been proposed which are more efficient than the simplex method [MATS85]. Routing in VLSI has been studied as a multicommodity problem in [TANG64].

6.1.2 Multicommodity Flows in MIN Transportation Networks

A set of requesting sources attempting to establish connections in a specific MIN can be formulated as a multicommodity problem. A small example suffices to illustrate an instance in which the transportation network cannot be solved as a single commodity problem. Sources *a, b, c, d, e, f* request destinations *a', b', c', d', e', f'* respectively. However, if the flow is indistinguishable the maximum flow is found to be 5, whereas the maximum of connection is found to be 4 (a flow from *c* to *d'* is realized but which does not form part of the input pattern).

The requesting input pattern and the MIN must undergo a transformation in order to be formulated as a multicommodity flow problem. Every distinct requesting inlet produces one unit of a distinct commodity (and produces no other type). Every requested destination demands one unit of a distinct type of commodity. Edges in the transportation network have unit capacity. All the active inlets and destinations (derived from the input pattern) will be nodes in the transportation network. Every link at which a conflict occurs between two or more inlets is represented by a node

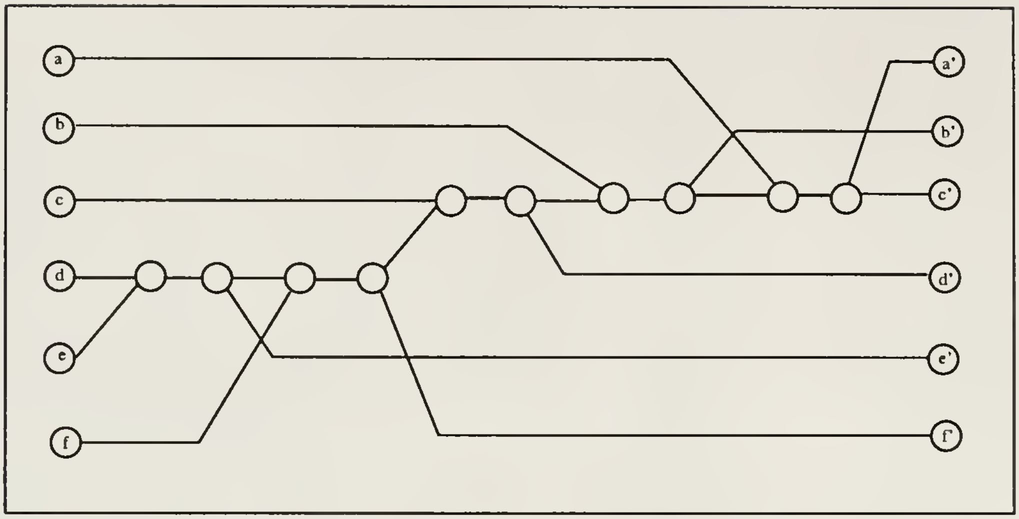

Figure 6.2. Path Setup is Not Single Commodity Problem

with two ingoing arcs and one outgoing arc (see Figure 6.2 for an example). The transportation network which is thus obtained contains no cycles and retains the unique path property of the original UPP-MIN.

By formulating the input pattern as a multicommidity problem, where the flows refer to messages sent from one node to another over communication channels (edges) of given capacities. The constraints placed on the problem is that the flows are integral, edges have unit capacity, and that each message is considered a single commodity.

In [NAGA89] some classes of multicommodity flow problems which have the max-flow min-cut property for directed networks also have the integral flow property.

We will show that by the criteria posed in [NAGA89], the optimum connections setup problem is guaranteed to have an integral optimum and which can be obtained by the simplex method.

Let Q be the class of transportation networks constructed from any input pattern and a given MIN topology. Any network N in Q can be described by a four tuple (G, P, d, c) where

- G is the finite directed graph with V nodes and E edges having a non-negative capacity.
- P is the set of K source-sink ordered pairs of commodities (messages).
- $d : \{1, \cdots, K\} \rightarrow Z^+$. Each $d(k)$ denotes the amount of supply (demand) of commodity k.
- $c : E \rightarrow Z^+$ is the capacity function of the edges.

In [NAGA89], two classes of networks are defined:

- Q is *capacity independent* if, for any network N in Q, the network $N' = (G, P, d, c')$ defined by:
  - $c'(i, j) = c(i, j) - 1$ and
  - $c'(i, j) = c(i, j)$ for all $(i, j) \in E - (i, j)$

  for an arbitrarily chosen $(i, j) \in E$ with $c(i, j) \geq 1$ also belongs to class Q.
- Q is *demand independent* if, for any network N in Q, the network $N' = (G, P, d', c)$ defined by:
  - $d'(k_1) = d(k_1) - 1$ and
  - $d'(k) = d(k)$ for all $k \neq k_1$

for an arbitrarily chosen $k_1 \in \{1, 2, \cdots, K\}$ with $d(k_1) \geq 1$ also belongs to Q.

Then Theorem 3.5 of [NAGA89] states that if a capacity and demand independent class Q has the max-flow min-cut property, then Q satisfies the integral flow property. (A network has the max-flow min-cut property if the cut condition holds for any cut of the network).

It can be shown that the multicommodity flow transportation network constructed from the input pattern will be both capacity and demand independent and thus will have an optimum integral flow. A linear program can be used to derive this optimum solution. This flow corresponds to the maximum connections set, as each unit of flow is contributed by a feasible shipment of a commodity from a source to its respective sink.

6.1.3 Maximum Setup Example

As an example, consider the MIN shown in Figure 6.3, and the input pattern given. Suppose that in a 8 × 8 Omega network, sources 0, 1, 2, 5 and 6 request destinations 6, 5, 5, 7 and 6 respectively. The bold lines in the figure indicate the interstage links which will be used during the transmission of this set of connections. The equivalent transport network is obtained by identifying the active inlets, switches and destinations, and Figure 6.4 reflects that. In this figure, inlet nodes by i, and destination/sink nodes by d. Next to each edge e we write $c(e), f(e)$ in this order.

The linear programming formulation of this problem becomes:

Maximize: $f = x_0 + x_1 + x_2 + x_5 + x_6$

Subject to: $x_0 + x_6 \leq 1$

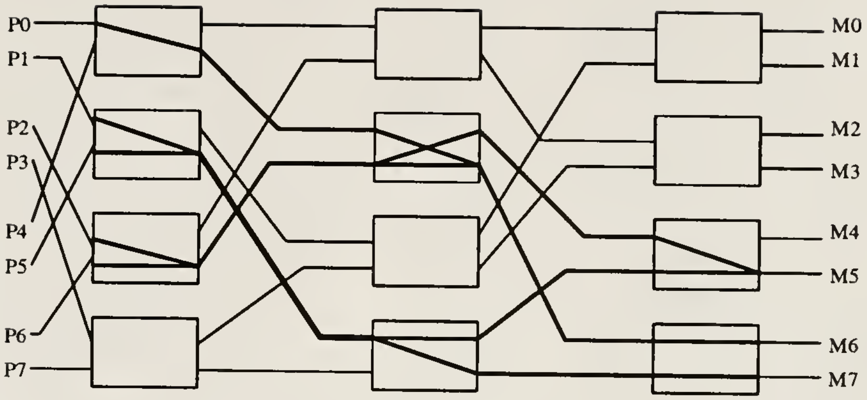

Figure 6.3. Input pattern in 8 × 8 Omega

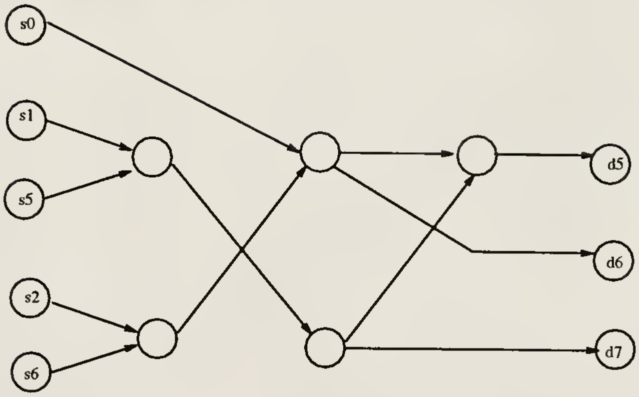

Figure 6.4. Transport Network

$$x_1 + x_5 \leq 1$$

$$x_1 + x_2 \leq 1$$

$$x_2 + x_6 \leq 1$$

An optimal solution is found to be $x_0 = 1, x_1 = 0, x_2 = 1, x_5 = 1, x_6 = 0$. Thus the optimal setup would be to grant inlets $0, 2,$ and 5 and to deny inlets 1 and 6.

6.2 The Re-Attempt Parallel Path Setup Method

The objective of this section is to describe a parallel path setup scheme for UPP networks. The objectives of any path setup method are first of all to establish the most number of paths and second to do this in a time and cost effective manner. The fastest method is without doubt the parallel method described in Section 6. However, an additional drawback of the parallel path setup scheme is that upon link conflict the arbitration is done in a random fashion. The victor of an arbitration in a particular stage may eventually lose in another round of arbitration in a latter stage. The question now becomes whether the connections which lost in earlier stages against latter stages victims could attempt to establish connection for another time.

The procedure below describes one method in which this could be done. The immediate advantage is obvious - there are definitely more connections being set up using this rationale than if otherwise left alone.

The design requires each network switch to implement a single buffer per output link. Upon a link conflict in the forward pass, the arbitration proceeds in the usual random fashion. The victim, however, instead of being dropped, will be buffered at the designated link from where it may attempt a second time if needed. The successful connections continue to contend in the following stages. Upon arrival at the destinations, the acknowledgment phase starts by sending positive acknowledgements along the links whose connections are granted and consequently negative acknowledgements travel back along links to sources which have been denied a connection. As these acknowledgements travel backwards two things can happen in any switch.

1. If the acknowledgement was positive, and there was a request residing in the buffer of that output link, then a negative acknowledgement should be generated from that point on and sent back (both positive and the newly generated negative acknowledgement now continue backward);

2. If the acknowledgement was negative, and there was a request residing in the buffer of that output link, this packet can attempt to set up a connection by traveling forward from the current stage onwards. One can single out potential losers by not transmitting if the requested destination is the same as the one which was negatively acknowledged to begin with.

The basic method above thus describes a iterative scheme in which acknowledgments being fed back into the network trigger setup requests to attempt connection after having been denied access earlier on. The arbitration scheme in this scheme is still random, an inherent drawback to the scheme. One immediate improvement to this scheme would be to enhance the arbitration scheme. By keeping track of the number of previous conflicts in earlier stages may reflect the desirability of granting such a request connection. A setup request with a lesser collision count will most likely not impede a great number of other connections to be estabished. This and another improvement to the basic scheme will be described and evaluated in the next sections.

The switch operations in the Re-APPS can be summarized as follows. A request is called *pending* if it has been routed to the next stage and no acknowledgement has yet been received. A path is *connected* if the switch receives a positive acknowledgement. The positive acknowledgement will propagate back through the incoming input link. An output link is available if there is no pending request nor connected path. When a request is rejected, a negative acknowledgement will be sent back on the incoming input link.

The *Re-Attempt Parallel Path Setup* (Re-APPS) algorithm proceeds as follows:

Algorithm 6.1 *Basic Re-APPS*

```
begin
    if two requests arrive
        get routing tags;
        if no conflict
            route the requests, the requests are pending;
        else
            invoke an arbitration procedure;
            route the winner, the request is pending;
            if the destination addresses are equal
                reject the victim;
            else
                buffer the victim at the output register;
    if one request arrives
        get routing tag;
        if the output link is available
            route the request, the request is pending;
        else
            if the destination address is equal to that of the pending request
                reject the request;
            else
```

```
                buffer the request at the output register;
    if a positive acknowledgement is received
        connect the path;
        if the output register is occupied
            reject the request;
    when a negative acknowledgement is received
        reject the request;
        if the output register is occupied
            route the request, the request is pending;
end Basic Re-APPS;
```

The switches in the last stages do not need to send the requests. They can acknowledge the requests based on conflict condition and the result of arbitration. In fact the switches in the second before last stage do not need to retransmit a buffered victim after a conflict. If the acknowledgement of the winner is positive, the victim will be rejected. Otherwise, the victim can be connected and a positive acknowledgement can be sent back.

It can be easily shown that one register per switch is enough to buffer a victim packet. Note that, when the register is occupied, the switch must have a pending request and has received two requests. Since there can have at most two pending requests in a switch which are routed to two different switches of the next stage, thus a switch will not receive any more request once the register is occupied. Also, when an acknowledgement (positive or negative) is received, the register is freed immediately.

A simple arbitration scheme can be used which selects the victim in a random fashion. An improvement to this scheme would be to enhance the method in which the arbitration is done. Certainly if one kept the count of previous conflicts as part of the setup request packet, a more likely preferred packet is one which has not suffered many collisions. In this way, the packet records how many victims it has made during its forward path setup. To resolve a conflict the victim is the packet whose collision number is the highest.

6.3 ReAPPS with Distributed Synchronization

In a synchronous transmission it can easily be seen that the transmission of the data packet depends on the time the last source is acknowledged. If there is no global knowledge a trivial manner would be to wait for the worst case setup. However, by combining the path setup, acknowledgement with a barrier synchronization mechanism the data packet transmission time can be earlier than in the worst case. The Re-APPS algorithm combined with an embedded synchronization mechanism is described below. For simplicity we only show the sending of synchronization signals and omit the details of the Re-APPS algorithm itself which was described in the previous section. Packets, with or without valid data, are assumed to be present at all input and output ports in order for this algorithm to work. An *activity* bit is used to indicate that the packet contains valid data.

The synchronization signals are assumed to be transmitted in a separate packet or piggybacked onto the request packets or acknowledgements. At the beginning of a path setup, all inlets send a forward synchronization (*FS*) to the switches in the first stage, irrespective of whether they have a request to setup or not. Every one of the switches in the last stage is responsible for sending back return synchronization signals (*RS*), when it receives *FS* signals at all of its input ports or when it is able to acknowledge both input ports (by receiving the signal from the destinations). At each input port of a switch, a DONE-PHASE-1 flag (*DP1*) will be set when a *FS* arrives. At each output port, a DONE-PHASE-2 flag (*DP2*) will be used to indicate the arrival of a *RS* signal.

<u>Algorithm 6.2</u> *Re-APPS with Embedded Synchronization*

```
begin
    when a FS signal arrives
        set DP1 flag;
        if both DP1 are set and the output register is free
            broadcast a FS signal to the next stage;
    when a RS signal arrives
        set DP2 flag;
        if both DP2 are set
            broadcast a RS signal back to the previous stage;
end Re-APPS with Embedded Synchronization;
```

It can be shown that the above synchronization algorithm guarantees that all sources receive a *RS* signal at the same time. Note that an arrival of a *FS* signal at an input port means that no more path setup request will be received through this input port. When a switch in the last stage receives both *FS* signals, all paths destined to it have been either connected or rejected. The switches in the last stage may issue the *RS* signals at different instances. However, the *RS* signals will be buffered at the output ports in the network except the last *RS* signal which will propagate back to the sources without any delay. Thus, all sources receive the last *RS* signal at the same instance and are synchronized to transmit data packets along connected paths.

It is interesting to know the possible delay of starting data transmission due to the synchronization algorithm. Let S_i be the source that is the last one of receiving

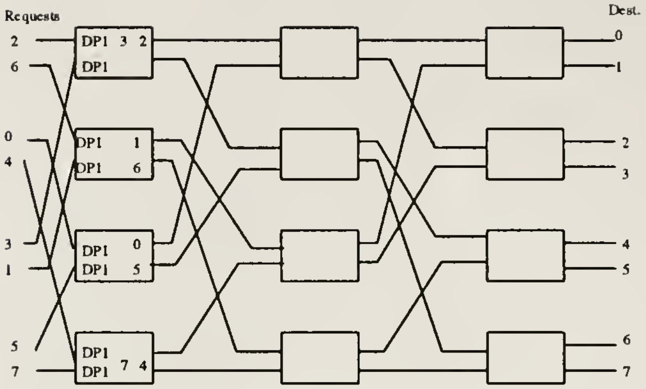

Figure 6.5. Embedded Synchronization Example

an acknowledgement (either positive or negative). The *FS* signal along the path of S_i's request will be blocked until the request is retransmitted or rejected. If the request is rejected, a negative acknowledgement will propagate backward, whereas the *FS* signal is sent forward. If the request is rejected due to a connected path in the first stage, S_i will receive the *RS* signal after the *FS* reaches the last stage and the *RS* signal propagates back. Thus, the worst case delay between the moment that S_i receives an acknowledgement and the moment that all sources receive the *RS* signal is less than the round trip delay of *FS* and *RS* signals.

We shall illustrate this algorithm by means of an example. Assume that in an 8 × 8 Omega network, sources 0, 1, 2, 3, 4, 5, 6, and 7 request destinations 2, 6, 0, 4, 3, 1, 5, and 7 respectively. Figure 6.5 reflects the first stage developments, where the *DP1* flags are set at all the first stage input ports since the inlets all sent a *FS* signal.

From Figure 6.5 it can be seen that the second and third switches in the first stage can broadcast a *FS* signal which sets the DP1 flag at the input ports in those switches. In the second stage, however, there is no switch that has received DP1 at all its input ports and hence cannot generate *FS* signal until that happens. Figure 6.6 shows the

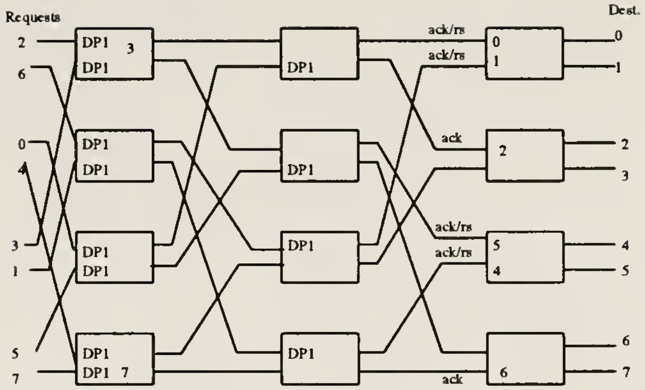

Figure 6.6. Embedded Synchronization Example Continued

progress of the packets once arrive at the input ports of the final stage. The first and third switches can generate an *RS* signal since they have received inputs at both the input ports. This signal then gets appended to the packets in the acknowledgement phase and sent back.

In Figure 6.7 the *RS* signals have set the *DP2* flags. Since there is no switch at this point which has the *DP2* flags set at both output ports, the signal cannot be propagated further. And since there are no buffered packets for retrial, the acknowledgements continue to the first stage. When these acknowledgements reach the first stage, and since all the acknowledgements were positive, the pending requests do not need to try to re-attempt. At this time, the remaining output ports in the first stage (in particular the first and last switches) are able to send a *FS* signal, setting the respective *DP1* flags in the second stage. If there were retrials, then the forward signal would be piggybacked onto the re-attempting request. However, in our example, we will make use of inactive packets and broadcasting to send the signals from stage to stage.

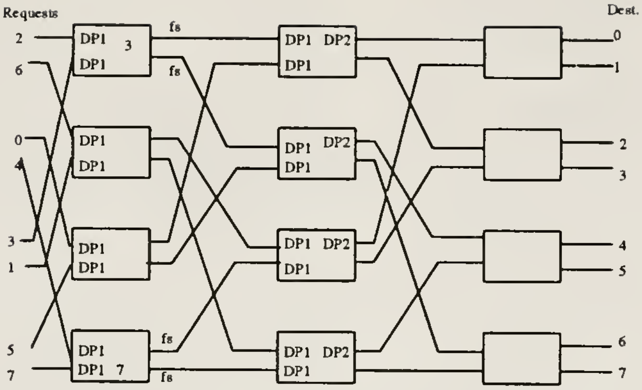

Figure 6.7. Embedded Synchronization Example Continued

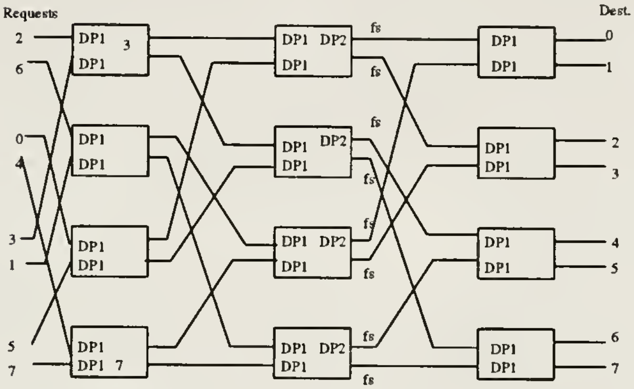

Figure 6.8. Embedded Synchronization Example Continued

All the switches in the second stage are then able to broadcast *FS* signals (see Figure 6.8) setting all the *DP1* flags of the final stage. This will cause that the remaining switches of the last stage send *RS* signals, and the final propagation of *RS* signals will set all remaining *DP2* flags and terminates the setup request. Since the switches operate synchronously, the switches in the first stage will all have their DP2 flag set simultaneously.

6.4 Analysis of Path Setup Delay of ReAPPS Scheme

The cost of the ReAPPS scheme is the overhead incurred in the setup phase. In this section we present the worst case analysis for the overhead incurred by the ReAPPS scheme. We assume that we have a $2^n \times 2^n$ banyan network with n stages.

The units of measurement will be setup cycles which are defined to be part of the data packet transmission cycle. One setup cycle is the time it takes to send a request from stage i to stage $i+1$, to arbitrate a possible collision in stage $i+1$, and to receive an acknowledgement message from stage $i+1$.

We will compare the overhead incurred by the ReAPPS scheme to Lea's IPS scheme which requires 2^n setup intervals. An interval here is defined to be the time needed to set up a source-destination pair. Lea's scheme is independent on the number of requesting sources, making the worst case equivalent to the average case. Even if there were no 2^n sources requesting connections the algorithm would still need this number of setup intervals since from individual sources' point of view the knowledge of other requesting sources is not available and hence the data transmission interval cannot proceed even though there are no more sources requesting.

We shall consider the worst case setup delay first. Let the stages be numbered from 1 to n where the switches in the first stage receive connection requests directly from the inlets. At a switch of stage i, we define *r(i)* to be the maximal number of setup cycles required to receive an acknowledgement from a switch of stage $i+1$ after a request is sent to the switch of stage $i+1$. It is clear that $r(n-1) = 1$. Assume that the retransmission of a collided request takes the same amount of setup cycles as the winner. Then, we have

$$r(i) = 1 + 2r(i+1)$$

for $i = 1, 2, .., n-2$.

The worst case is thus the case in which retransmission still occurs from the first stage, yielding thus:

$$\begin{aligned} max(Setup\_Delay) &= 1 + 2r(1) \\ &= \sum_{i=0}^{n-2} 2^i + 2^{n-1} r(n-1) \\ &= 2^n - 1 \end{aligned}$$

The overhead in terms of bits can be computed under the assumption that request packets are of fixed length and that time is divided into slots where each slot consists of a path setup interval and a data transmission interval as illustrated in Figure 6.9.

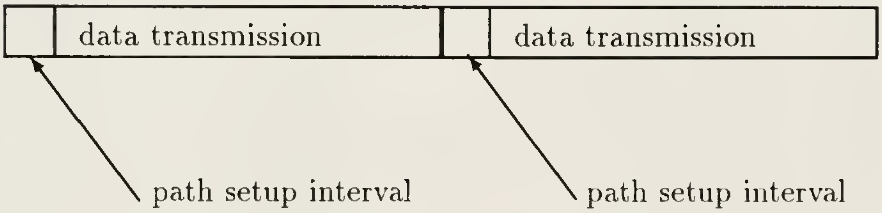

Figure 6.9. Time slot in a nonbuffered packet switch banyan network

The path setup overhead cannot assumed to be zero when computing the throughput of the network. If *Throughput* is the value obtained when overhead is assumed

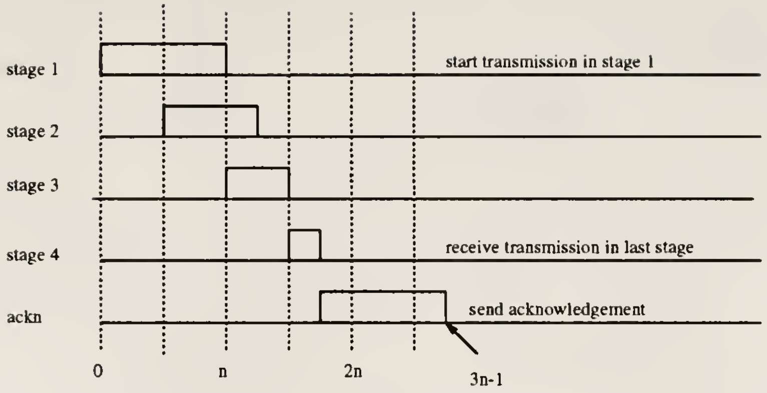
stage 1
stage 2
stage 3
stage 4
ackn
start transmission in stage 1
receive transmission in last stage
send acknowledgement
0
n
2n
3n-1

Figure 6.10. Transmission of request in a 4-stage banyan network

to be 0, then the actual throughput is equal to $Throughput \times \frac{L}{L+M}$ where L and M are packet length and path setup interval length in bits, respectively. M is in the order of $3n$ bits, where n is the number of stages [LEA87] - Figure 6.10 illustrates this for a 4 stage banyan. The assumption is made that at each stage the first bit is removed and the rest is transmitted to the next stage. At each stage the delay is 2 bit transmission time (one for the removal and another for the input port to output port transmission). Acknowledgements are assumed to be single bit.

Pure IPS (setup intervals = number of requesting sources) would then have a multiplicative factor of $\frac{L}{L+N.M}$ if there are N requesting sources. The total setup time for pure IPS is $2^n.3n$

In the worst case ReAPPS requires $3n$ bit transmission time for the first pass and $2^n - 1$ bit transmission for re-attempts which yields a total of $2^n + 3n - 1$ bit transmission time. The path setup time for several system sizes in an unbuffered banyan network for several packet lengths are given in Table 6.1 below.

6.5 Performance Evaluation of Re-APPS

In order to evaluate the performance of the Re-APPS scheme, we have conducted simulations of the Re-APPS and provide comparison results with the optimum throughput, the PPS and IPS schemes.

The simulation model assumes an Omega network of varying size. According to a given request rate, inlets generate requests according to a traffic pattern. The input pattern can be a permutation, the traffic pattern can be uniformly distributed, or contain a single hotspot.

In the discussion below, unless otherwise stated, the size of the network is assumed to be 64 × 64, the load assumed to be 1.0, with the number of setup periods measured to be 2000. During each setup period a new and independent set of requests is generated, attempts connection, and those which were unsuccessful are dropped.

In Table 6.2 the percentage of connections granted is given for uniform traffic, under varying load. Table 6.3 shows the same information, but for permutation traffic. As expected, the improvement is mostly for high loads, for the probability of having a conflict each stage is higher. In comparison, permutation traffic has a higher throughput, since output conflicts are eliminated by ensuring the input patterns have distinct destinations. Due to the internal blocking, *PPS* is limited to 44% throughput, whereas the Re-APPS is able to improve this to 52%.

Table 6.4 gives the performance of the three schemes for permutation traffic, and various network sizes. The optimal throughput values are included for comparison reasons. It can be seen that the Re-APPS with collision count approaches the optimal throughput within 3%. The throughput decreases drastically with the size of the network, in accordance with Patel's [PATE81] findings. The IPS improves on this throughput, but the Re-APPS obtains consistently a better performance than the IPS scheme. The advantages of the Re-APPS scheme is clearly that at a far smaller cost than the IPS scheme, a better throughput can be obtained.

6.6 Chapter Summary

The maximum path setup problem can be formulated as a transportation network which can be proven to have an integral flow. This implies that this flow can be found using any linear program.

ReAPPS is a newly proposed setup mechanism which, at a slight addition in overhead, can outperform both the PPS and the IPS scheme. The number of connections granted by ReAPPS approaches the optimum within 3%.

Packet Length	*System Size*	*PPS*	*ReAPPS*	*IPS*
500	16 × 16	2.4%	5.40%	38.4%
2000	16 × 16	0.6%	1.35%	9.6%
2000	64 × 64	0.9%	4.05%	57.6%

Table 6.1. Setup Overhead for PPS, IPS and ReAPPS schemes

Load	*PPS*	*IPS*	*Re-APPS-Basic*	*Re-APPS-Coll*
0.1	0.86	0.87	0.87	0.88
0.2	0.76	0.77	0.78	0.79
0.3	0.67	0.69	0.70	0.71
0.4	0.61	0.63	0.64	0.66
0.5	0.55	0.58	0.59	0.61
0.6	0.50	0.53	0.55	0.57
0.7	0.45	0.50	0.51	0.53
0.8	0.42	0.46	0.48	0.50
0.9	0.39	0.44	0.45	0.47
1.0	0.36	0.42	0.42	0.44

Table 6.2. Various Schemes under Uniform Traffic

Load	*PPS*	*IPS*	*Re-APPS/Basic*	*Re-APPS/Coll*
0.1	0.90	0.90	0.90	0.91
0.2	0.82	0.83	0.83	0.84
0.3	0.75	0.76	0.77	0.78
0.4	0.69	0.70	0.71	0.73
0.5	0.63	0.65	0.67	0.69
0.6	0.58	0.61	0.63	0.64
0.7	0.54	0.58	0.59	0.61
0.8	0.50	0.54	0.55	0.58
0.9	0.47	0.52	0.53	0.55
1.0	0.44	0.49	0.50	0.52

Table 6.3. Various Schemes under Permutation Traffic

Size	*PPS*	*IPS*	*Re-APPS/Coll*	*Optimum*
8 × 8	0.68	0.70	0.73	0.735
16 × 16	0.58	0.61	0.65	0.670
32 × 32	0.51	0.54	0.58	0.604
64 × 64	0.44	0.49	0.52	0.547
128 × 128	0.40	0.45	0.48	0.509
256 × 256	0.36	0.41	0.44	0.470
512 × 512	0.33	0.39	0.41	0.442
1024 × 1024	0.30	0.36	0.39	0.424

Table 6.4. Effect of Network Size on Setup Schemes

CHAPTER 7
IMPROVING THE PERFORMANCE OF FAST PACKET SWITCHES

Packet switching and stochastic multiplexing have been recognized as the most appropriate approaches to achieve high speed communication and high resource utilization. Due to the stochastic nature of arriving packets, there may be more than one packet from different input ports destined to the same output port simultaneously. One of the contending packet can be delivered to the output stream. The other packets can be saved in a queueing buffer and wait for an available transmission slot. Queueing buffers can be implemented in the input port of the switch fabrics or in the output port, or in both ports. Nonblocking fast packet switches can thus be classified as being either input or output queueing, with a speedup factor of S where S is the number of packets which the switch can deliver to the same output port simultaneously. Input queues are designed to store packets which cannot be routed immediately through the fabric, and output queues store packets which need to wait for the service of the output links. Performance evaluations on these two queueing schemes show that, with infinite buffers, the throughput of input queueing scheme is limited to only 58% due to *head-of-line (HOL)* blocking, whereas the output queueing scheme can sustain a load up to 100% [KARO87, HLUC88, ILIA90]. However, to utilize the output queues, all contending packets must be able to join the destined queue simultaneously. With non-blocking switch fabrics, in order to achieve this, either the switch speed must be increased or multiple paths of entering the queue must be provided. This requirement inevitably increases the complexity of the switch fabrics even if a modest speedup factor of 4 can lead to a 99% throughput as demonstrated in [CHEN91].

To improve the performance of input queueing schemes, several reservation approaches have been suggested. The token ring principle is adopted in [BING88] where a stream of tokens is passed through all input ports to perform output reservations. Arbitrators as well as contention resolution devices can be added in conjunction with the switch fabrics [ARAK90, CISN90, KARO87]. These devices accept the transmission requests, the transmission requests, schedule the transmissions, and then acknowledge back to the input ports whether the packet transmission can be initiated. Though these approaches introduce different degrees of complexity or cause a limitation on the speed at which the switch can be operated, the performance improvements are less satisfactory.

Thus it can be seen that an important implementation issue in the design of a fast packet switch is the development of a scheduling algorithm among the input ports which may have diverse service requirements and arrival rates. Scheduling becomes an important concern when a large number of slow rate input ports need to communicate to a smaller number of high speed outlets. In order to increase the utilization of the fast output links, the switch can multiplex the input and present the combined input rates to the outlets. Moreover, in telecommunications systems, the type of offered load to the switch depends varies with time, and hence must be considered when designing schemes which will provide an improved throughput and delay. The most common scheduling discipline is FIFO, where the queues service messages according to the head-of-the-line discipline. This chapter studies the manipulation of the input load in order to increase fast packet switches' throughput by an arbitration policy that decides the order in which packets at various switch input ports will be serviced. The method and its parameters are thoroughly measured and found to improve input queued packet switches, in particular switches based on the banyan switch fabric.

This chapter is organized as follows. Section 7.1 describes the switch architecture and its operation. The scheduling algorithm is also presented. Section 7.2 details the analysis performed on the CWC switch, under uniform traffic. Section 7.3 presents empirical results obtained on the CWC switch under uniform and hot spot traffic patterns. Section 7.4 presents a summary of the work described in this chapter.

7.1 The Design of Circular Window Control (CWC) Scheme

A generic packet switch is a switch which routes packets from M inputs to N outputs, where we will make the assumption that all the lines have the same transmission capacity. We also assume that packets are of the same size, that time is slotted with the slot size being equal to the transmission time of a packet on a a line, and that the slot boundaries on all input lines are synchronized, in other words, the operation of the switch is synchronous.

An input queue switch architecture is shown in Figure 2.4, which contains buffer space at each inlet to the switch. When a packet arrives it is placed in the buffer space provided there is space, and awaits access to the switch fabric. When the queues are served in a FIFO manner, only packets at the head of the input queues are considered for transmission during the current transmission slot. When contention occurs, all the contending packets, except the winner, must be retransmitted during later slots, while blocking packets behind the head of queues which may be destined to idle outputs or could be transmitted through idle routes. It is this type of head-of-line blocking which limits the throughput of input queueing schemes.

There are several ways in which the throughput may be improved. First, the throughput may be increased by dropping packets, if the arrival rate is sufficiently high. The dropping of packets reduces input queue blocking, but can also lead to throughput reduction if the utilization of the input trunks are small, for new packets do not arrive fast enough to replace the dropped ones. Another way is by relaxing the FIFO order of the input queues. This allows packets other than the one at the head of the queue to be serviced.

One manner in which to relax the FIFO order is to select packets from input queues with the objective to minimize the chance of contention. This process can be done dynamically by comparing the destinations of the packets at the heads of all non-empty queues. It may require having to look into a number of packets within a predetermined checking depth. This imposes a difficult and complex implementation problem no matter whether the process is done in a centralized or distributed manner.

7.1.1 Switch Architecture and Operation

A static approach to scheduling packets in a non-FIFO order, called circular window control (CWC) scheme, is discussed in this section. Consider an $N \times N$ switch.

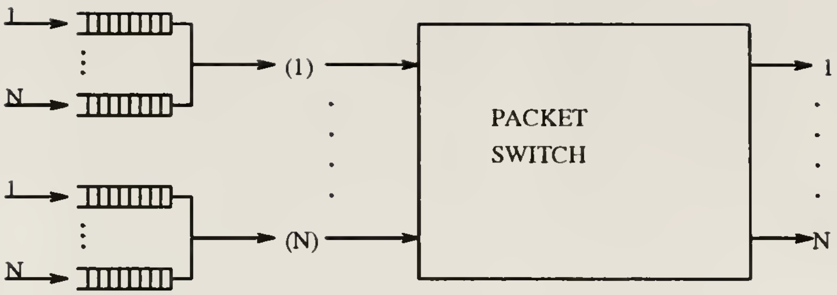

Figure 7.1. CWC Architecture Model

Assume that each input port of the switch has N subqueues which are dedicated to packets of the same destination as shown in Figure 7.1. The packets arriving at each input port join the subqueues in accordance with their destinations, and within the same destination they are queued based on their arrival times (i.e. FIFO order). We group w consecutive transmission slots as a *window* and N consecutive windows as a *frame*. The frames of all input ports are synchronized. Within a frame, the CWC scheme selects one of the subqueues to be the primary subqueue for transmission during a window. The selection is done in a sequential fashion and started from distinct subqueues at all input ports. Thus, a transmission window is circulating between the subqueues at an input port. During each slot, the windows at all input port are located at subqueues designated to different outlets. When the packets transmitted are from primary subqueues of all input ports, the CWC scheme guarantees the pattern causes no output contention. Furthermore, if the i-th input port starts the primary queue from the i-th subqueue, the packets from the primary queues form a circular permutation which is passable in a banyan-based multistage interconnection network and lead to contention-free transmissions.

The CWC scheme is detailed in Figure 7.2. During the current window, each inlet will attempt to transmit from their primary subqueue, but if the subqueue is empty, an alternate subqueue is chosen according to several strategies with the risk of having an output conflict. Potential collision exist in this manner however, for less than full load this may compensate for loss of bandwidth.

The choice of an alternate subqueue must be meticulous, since it is well known that not all permutations are passable in a Banyan network. The CWC scheme guarantees that the input patterns will be cyclically sorted, which forms the basis of the sorting networks and have been proven to be passable in Banyan networks. Selecting from a random queue may impede one or more packets from primary queues to be blocked. Though this can be resolved by prioritizing primary queue packets, it defeats the purpose of an alternate queue trying to compensate for loss of bandwidth. Hence of the alternate queue strategies, one may expect that the *random* and *longest queue* strategies may not yield significant improvement over a "no alternate queue" strategy. The *Topological Neighbor (TN)* is defined to be the queue containing the destination address which share the same first stage switch. This choice implies that that the packet can at most cause a collision in the first stage, but no where else. The TN is topology dependent, for example in an Omega network the TN of inlet i is $i + \frac{N}{2}$ assuming N is the number of inlets.

The advantage of the CWC scheme is that for high loads the input patterns presented to the switch are sorted. This avoids the need for a sorting, and trap network which typically add large delays to the packets' transmission. The disadvantage is

```
for (inlet = 0; inlet < N; inlet++) {
    subqueue = (inlet + ( ( current-time / window-size ) MOD N ) MOD N;
    if (subqueue != EMPTY) {
    // transmission will be from the primary window queue //
       consider head-of-queue for transmission;
    } else { // obtain another subqueue //
    // transmission from a secondary queue //
       STRATEGY-1: choose a random queue;
       STRATEGY-2: choose the longest queue;
       STRATEGY-3: choose the topological neighbor;
    }
```

Figure 7.2. The Circular Window Control Scheme

that it considers only one single queue (checking depth of one), which may lead to low utilization, especially when the load is low.

The algorithm can further specify a priority to be given to packets from primary subqueues or from alternate subqueues. The priority will help control congestion by assigning a higher priority to packets from longer queues, thus reducing the probability of dropping packets when the queue length is finite, but making no difference on the overall throughput. This may not be the case in blocking switches such as a Banyan and the highest priority in this case is often given to the packet from primary queues.

A CWC example is given in Figure 7.3 where the primary subqueues are shown in a circular fashion of 0, 1, 2 and 3. The window size here is assumed to be 2. The shaded areas indicate the packets which are selected from alternate subqueues while the primary subqueues are empty during these transmission slots. It can also be seen that the throughput can be enhanced by selecting packets from alternate subqueues rather than not transmitting any since even non-primary packets may result in a nonconflicting transmission.

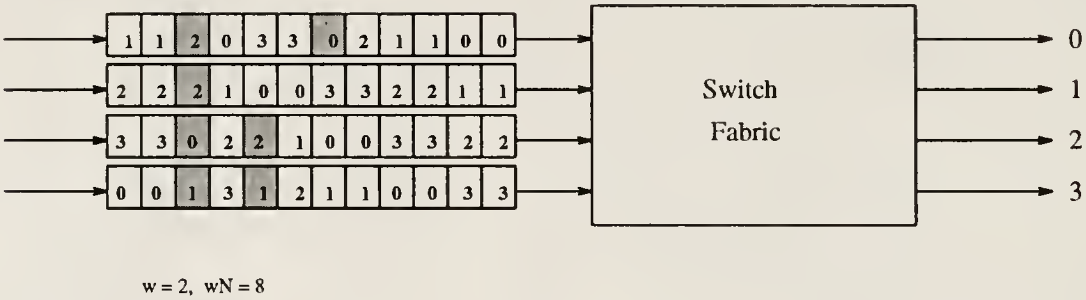

Figure 7.3. An Illustrating CWC Example

To implement the CWC scheme, each inlet must be able to examine the destination of arriving packets and place them in the corresponding subqueues. The subqueues can be arranged in physically separate buffers or share the same buffer space. In the latter case, a threshold on the maximum space a subqueue can occupy should be established. This can prevent the possible buffer starvation due to a flood of packets destined to a single outlet.

7.2 Analytic Models for the CWC Scheme

In the following, we present the performance evaluations of the CWC scheme. We shall first show that the maximum throughput under the CWC scheme can reach 100% under a uniformly distributed traffic and with infinite buffers. Then, we turn to a schedulability analysis of the CWC scheme. The schedulability is measured by considering only the arrivals within a fixed size frame. No queueing is incorporated across the frames. The comparisons of a perfect scheduling algorithm, a random selection scheduling and the CWC scheme are provided.

7.2.1 Throughput Analysis

Assume that there are infinite buffers at each inlet and the traffic is uniformly distributed where all destinations are requested with an equal probability. The switch is under a full load condition, that a packet arrives at each inlet per each transmission slot. The interesting measure is the maximum throughput under the CWC scheme.

Let all subqueues be empty initially. Let $M_{ij}(k)$ denote the number of packets transmitted up to and including the k-th frame from the j-th subqueue of the i-th inlet. Due to the symmetric nature, we may use the abbreviation $M(k)$ to represent $M_{ij}(k)$. The throughput at each output port is then

$$\theta = \lim_{k\to\infty} \frac{NM(k)}{kwN}$$

Consider two revised schemes. The first one is similar to the CWC except that it transmits no packet if the primary subqueue is empty. The second one is similar to the first one, except it considers only the packets arrived before the current frame for transmission. The packets arrived during the current frame will be delayed. Let $M'(k)$ and $M''(k)$ be the number of packets transmitted from a subqueue up to the k-th frame under these two schemes respectively. Clearly, $M(k) \geq M'(k) \geq M''(k)$ for all $k > 0$. Under the second scheme, denote $a(k)$ as the number of packets arrived at a subqueue during the k-th frame. $a(k)$ follows the Binomial distribution with parameters wN and $1/N$ for all k. The subqueue length at the end of the k-th frame, $q(k)$, can be specified as the following Markov chain:

$$q(k+1) = \begin{cases} q(k) + a(k+1) - w & \text{if } q(k) \geq w \\ a(k+1) & \text{otherwise} \end{cases}$$

With the mean of $a(k)$ equals to w, the balance equation of the above Markov chain has no stationary solution and all states are transient. Thus, $Prob\{q(k) < w\} = 0$ as $k \longrightarrow \infty$ and $\lim_{k \to \infty} M''(k) = wk$ with probability 1. This implies that $\theta'' = \theta' = \theta = 100\%$.

Note that this maximum throughput is achievable in both nonblocking switches and any banyan-based switches, as long as the packet permutations formed by the primary subqueues are passable through the switches. In addition, this maximum throughput is not affected by the strategy of selecting an alternate subqueue when a primary subqueue is empty.

7.2.2 Schedulability Analysis of the CWC Scheme

The CWC scheme basically provides a mix of deterministic and stochastic scheduling during a frame. The deterministic part is incurred when the primary subqueue is not empty, otherwise, the scheme takes a stochastic approach to select an alternate subqueue. In order to evaluate the schedulability of the CWC scheme, we examine the average throughput of a non-blocking switch under the following switch system. The switch system has a full load such that, during every transmission slot, there is a packet arriving at each inlet. The packets arrived during the current frame are batched and are transmitted in the next frame. If a packet cannot be delivered in the next frame, it is lost. Thus, we explicitly exclude the effect of queues. It is also assumed that the packet arrival processes at all inlets are i.i.d. and the packets' destinations are uniformly distributed among all outlets. The probability of having i packets destined to a specific outlet among K arrivals is then:

$$\alpha_i(K) = \left(\begin{array}{c} K \\ \\ i \end{array} \right) (\frac{1}{N})^i (1 - \frac{1}{N})^{K-i}$$

where $0 \leq i \leq K$.

Three scheduling schemes are considered here: a perfect scheduling algorithm, a random selection scheduling, and the CWC scheme. The perfect scheduling algorithm is assumed to be capable of transmitting up to wN packets per outlet. It cannot reach a 100% throughput since, with the random arrivals, there may be less than wN packets destined to a single outlet. Thus, the throughput at a single outlet is:

$$\theta_p = \frac{1}{wN} (\sum_{i=0}^{wN-1} i\alpha_i(wN^2) + wN \sum_{i=wN}^{wN^2} \alpha_i(wN^2))$$

Let $\beta_i(K) = \sum_{j=0}^{i} \alpha_j(K)$. The above throughput can be formulated as

$$\theta_p = 1 - \frac{1}{wN} \sum_{i=0}^{wN-1} \beta_i(wN^2)$$

When the random selection scheduling algorithm is used, each inlet selects a packet randomly to transmit. If there is an output contention, the loser will be dropped. Thus, based on the analysis in [PATE81], the throughput per outlet is

$$\theta_r = 1 - (1 - \frac{1}{N})^N$$

The average throughput of CWC scheme under this operation model can be approximated by examining the successful transmissions during a window at an inlet. Assume that the priority is given to the packets transmitted from the primary queues. [1] In a given window with a size of w slots, w packets are guaranteed to be transmitted if the corresponding subqueue holds more than or equal to w packets. If the subqueue has less than w packets, the inlet will select some packets destined to other outlets. An alternate packet can be successfully transmitted only if the slot is neither used by the packet from a primary subqueue dedicated to the same outlet during the current window, nor other alternate packets are destined to the same outlet.

Thus, the average number of packets transmitted successfully during a window at an inlet can be approximated by:

$$w \sum_{i=w}^{wN} \alpha_i(wN) + \sum_{i=0}^{w-1} i\alpha_i(wN) + \sum_{i=1}^{w-1} \alpha_i(wN) \sum_{j=i}^{w-1} \sum_{k=0}^{j} \alpha_k(wN)$$

where the first two terms count the packets from the primary subqueue and the last term is for the possible transmissions from alternate subqueues. Thus the average throughput is:

[1] This assumption does not affect the throughput measure, but makes the traffic analysis at each inlet simple.

$$\begin{aligned}\theta_c &= \frac{1}{w}(w\sum_{i=w}^{wN}\alpha_i(wN) + \sum_{i=0}^{w-1}\alpha_i(wN)(i + \sum_{j=i}^{w-1}\sum_{k=0}^{j}\alpha_k(wN))) \\ &= 1 - \frac{1}{w}\sum_{i=0}^{w-1}\beta_i(wN) + \frac{1}{w}\sum_{i=0}^{w-1}(\beta_i(wN))^2 \quad .\end{aligned}$$

A numerical comparison of the throughputs is provided in Table 7.1 with different values of w and N. In the table, we also show the simulation results of the throughput under the CWC scheme. The error between the approximation and simulated model is less than 3%. These numerical results illustrate that the schedulability of the CWC scheme is much higher than that of a random selection scheme and is comparative to a perfect scheduling algorithm.

7.3 Performance Evaluation

In addition to the maximum throughput analysis and schedulability evaluation, we have performed simulations for the CWC scheme for various conditions. We assume that at each timeslot the inputs will present their packets of which a subset will be allowed to transmit. We will assume no speedup and that each inlet can present at most one packet per timeslot. We measured two types of traffic, *output concentration* or *hot spot*, and *uniform* traffic. The impact on these switches imposed by these traffic loads is presented, with varying arrival rates. Several strategies for selecting packets from alternate subqueues are discussed and results are compared. The main measures we are interested in are throughput and mean waiting time under finite

Window	*N*	*Perfect*	*Random*	*CWC Analysis*	*CWC Simulation*
1	2	0.812	0.750	0.812	0.810
1	4	0.831	0.684	0.784	0.774
1	8	0.869	0.656	0.774	0.758
1	16	0.904	0.644	0.771	0.751
1	32	0.931	0.638	0.769	0.749
1	64	0.951	0.635	0.768	0.747
2	2	0.863	0.750	0.863	0.862
2	4	0.879	0.684	0.839	0.831
2	8	0.907	0.656	0.829	0.817
2	16	0.932	0.644	0.825	0.811
2	32	0.951	0.638	0.823	0.809
2	64	0.965	0.635	0.822	0.808
4	2	0.902	0.750	0.902	0.901
4	4	0.914	0.684	0.882	0.877
4	8	0.934	0.656	0.875	0.866
4	16	0.952	0.644	0.871	0.862
4	32	0.965	0.638	0.870	0.860
4	64	0.975	0.635	0.869	0.859

Table 7.1. Schedulability Analysis

length queues and shared queues in banyan switches. Where appropriate, we present results obtained for the CWC schemes in crossbar switches as well. For all results reported in the following, the confidence interval is within 5% at 95% confidence level.

Although the maximum throughput of CWC scheme can reach 100% when the queue length is infinite, packets will be blocked if the queue length is finite. With a crossbar switch, Figure 7.4 shows the throughput of the CWC scheme under various buffer sizes compared to pure input and pure output queueing switch systems. The switch size is 64×64. The queue length is assumed to be infinite in both pure input and output queueing switches. The CWC switching systems considered have finite subqueues of size 2, 4, and 8 at each inlet respectively. The traffic is assumed to be uniform and the selection of an alternate queue is random. Under these conditions, it can be seen that the CWC switching systems have throughputs drastically better than the input queued switches. The throughput of the CWC scheme approaches the output queued scheme as the buffer size increases.

When a blocking fabric such as the banyan is used, the CWC can improve the throughput by eliminating the head-of-line blocking and switch conflict. Figure 7.5 shows the throughput of the CWC scheme implemented for banyan switches of sizes 16 and 64, with shared or dedicated buffers. The number of buffer spaces at each subqueue is 2 for dedicated buffers at each inlet. With shared buffers, each inlet has a total 32 and 128 buffers for the switches of size 16 and 64, respectively. Under the CWC scheme, the source-destination permutations are passable in the banyan switch if all inlets choose the packets from their primary subqueues. Thus, no switch conflict will occur. It can be seen that even with subqueue lengths of 2 a significant improvement can be obtained over the pure input queued banyan. Due to head-of-line blocking, the pure input queued banyan fabric reaches saturation at a load of approximately 0.4. The output queueing approach is ignored here since the approach requires that all packets be able to join their destination outlets simultaneously and no switch conflict can be allowed.

In Figure 7.6, we present the mean waiting time for CWC banyan switches with infinite and finite queues. The mean waiting time under the pure input queued scheme is compared to that of the CWC scheme in the figure. The figure shows that the CWC scheme provides a good tradeoff in terms of throughput obtained and mean waiting time. Though the CWC architecture implements an input queue, the mean delay is much less than that of a FIFO input queue switch of the same size. When the input load is light, a transmission of packets from an alternate subqueue may not cause a contention since the corresponding primary subqueue of another inlet is likely to be empty. On the other hand, if the load is heavy, most packets are selected from the primary subqueues which again incur no contention. Thus, the regulated traffic under the CWC scheme produces much less contentions. Existing contentions and window circulation cause the CWC to incur this longer delay.

Another advantage of the CWC scheme is that traffic patterns which are otherwise detrimental in switching systems, such as the “hot spot”, can be tolerated by the CWC scheme. By eliminating the head-of-line blocking, CWC switched systems are able to select packets destined to outlets other than the heavily requested destinations. Having shared queues will eventually saturate all the buffer capacity with hot messages, and cause the throughput to reach a definite saturation. Dedicated queues will cause many hot messages to be dropped due to the early saturation of their queues. This is reflected in Figure 7.7, where CWC banyan switches of 64 × 64 with buffer capacity of 128 buffers per inlet (dedicated or shared), under uniform or hot spot traffic. We assumed that 0.02% of all arriving packets are destined to a hot spot. With an arrival rate larger that 78%, the hot spot will be saturated. Figure 7.8 shows the delay incurred by these schemes and the imposed traffic. The mean delay of the shared queue schemes is significantly higher due to the saturation of the shared queues with the packets destined to the hot spot, causing not only more conflicts but also causing non-hot messages to be dropped due to buffer overflow.

From the above simulation results, we believe that the performance of the CWC scheme can be improved in several ways. The results shown until now, assumed that alternate packets are selected in a random fashion. Selection of packets from alternate queues can be done in different ways. In our simulation we have compared the random selection strategies with the *longest subqueue with no priority*, *longest subqueue with priority* and the *topological neighbor* strategies. Choosing from the longest queue has as an immediate advantage that the flow is controlled at the same time and the possible blocking due to insufficient buffer space is reduced. Moreover, priority can be given to the packet selected from a longer subqueue when a switch contention occurs. The relative performance of the alternate queue strategies is depicted in Figure 7.9

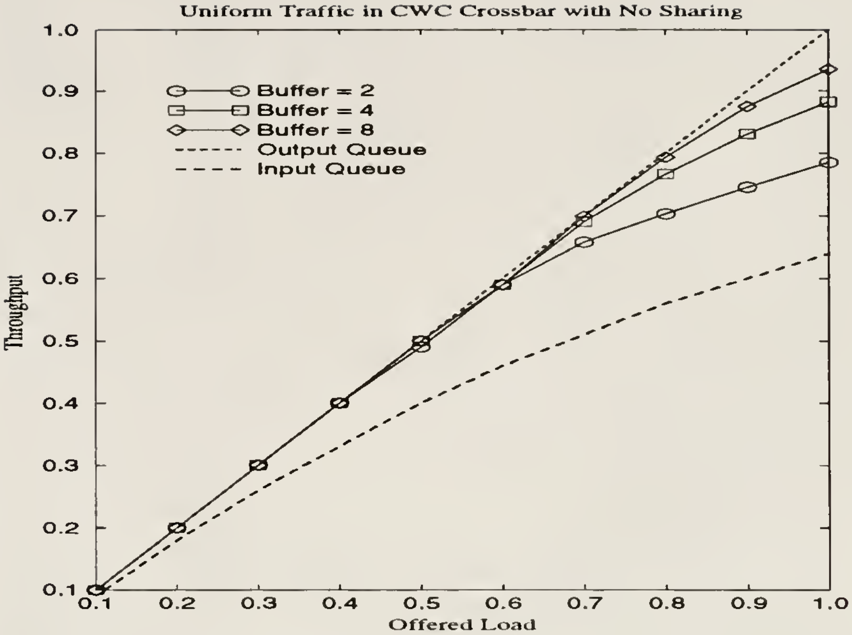

Figure 7.4. Throughput of CWC crossbar switch systems

This will reduce the blocking probability as well as the probability of having an empty primary subqueue. Thus, the probability of having output contentions will also be smaller.

The window size in our experiments was kept at one. Figure 7.10 provides the throughput of a 16 × 16 switch with window sizes of 1, 2, and 4 respectively. Larger window sizes have an effect when the load is intermediate and cause a modest increase in the mean waiting time when the subqueues are assumed to be infinite. This moderate difference justifies our experiments which were kept at a window size of one.

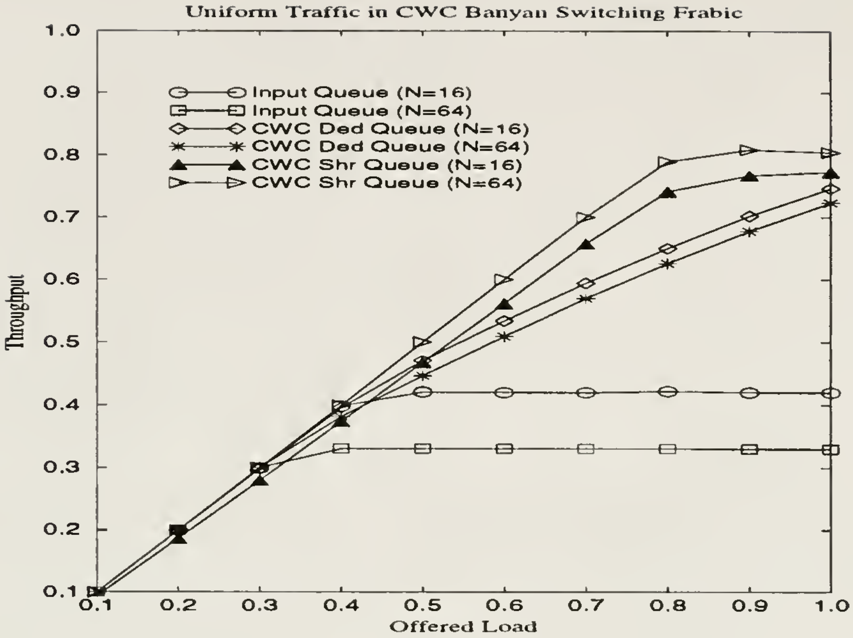

Figure 7.5. Throughput of CWC banyan switch systems

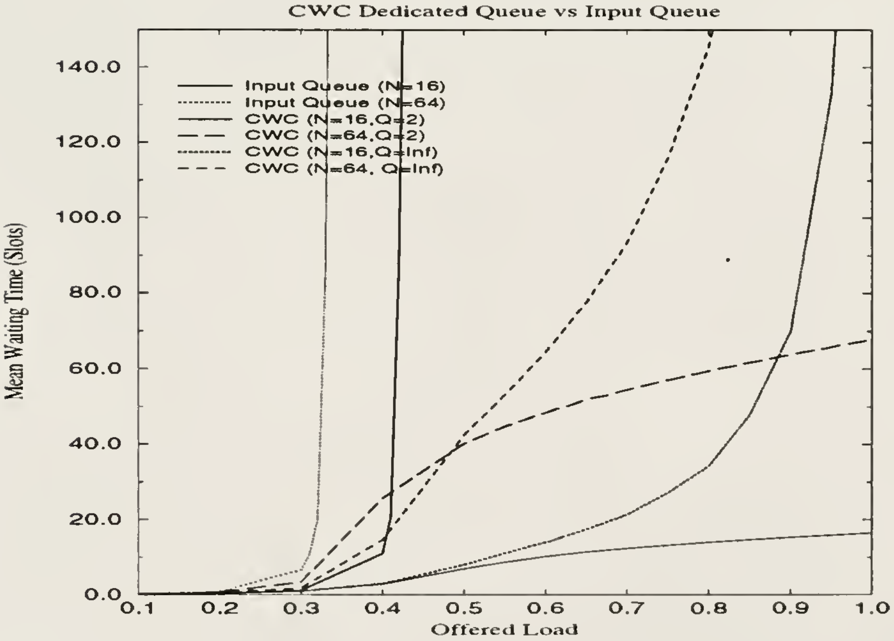

Figure 7.6. Mean waiting times of CWC scheme vs. input queueing scheme

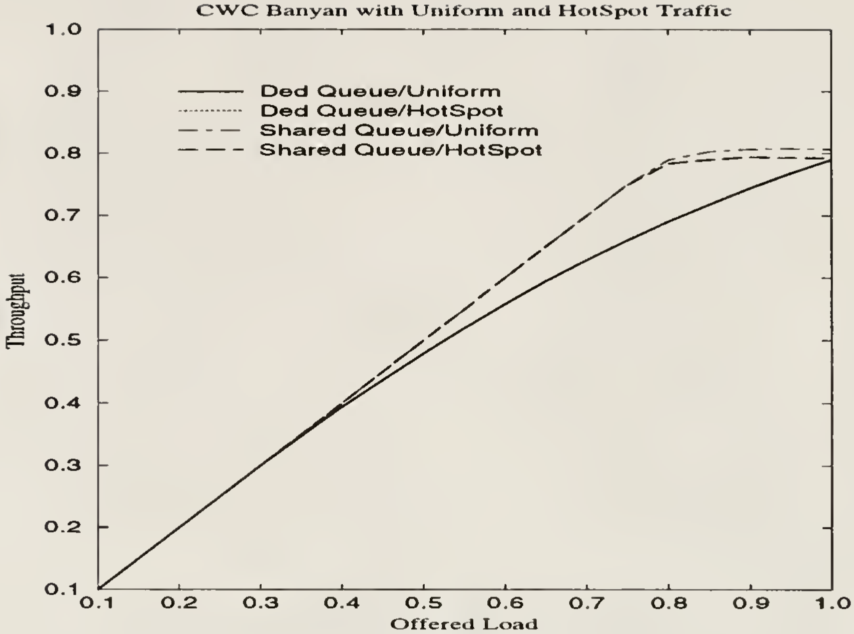

Figure 7.7. Throughput of CWC schemes under Uniform and HotSpot traffic

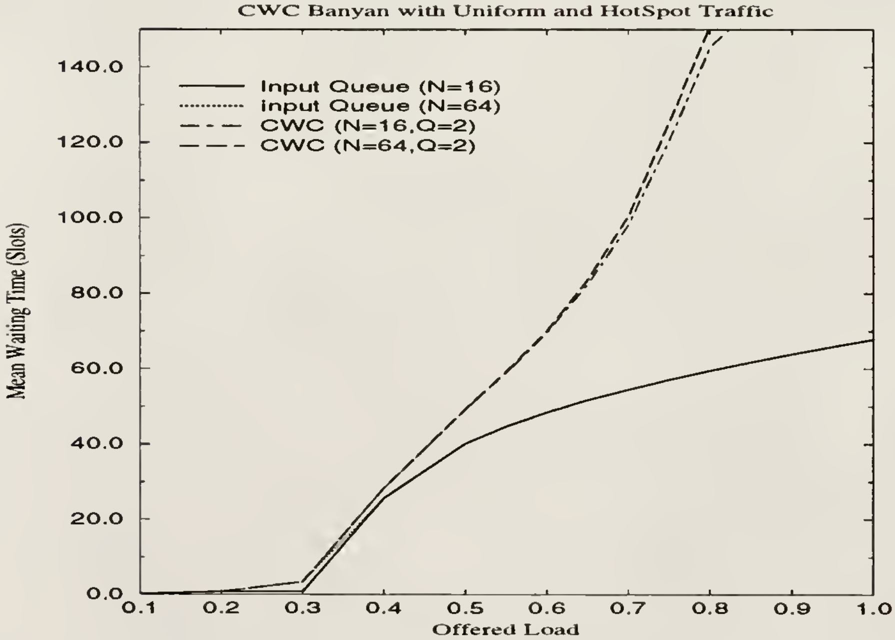

Figure 7.8. Mean Waiting time of CWC schemes under Uniform and HotSpot traffic

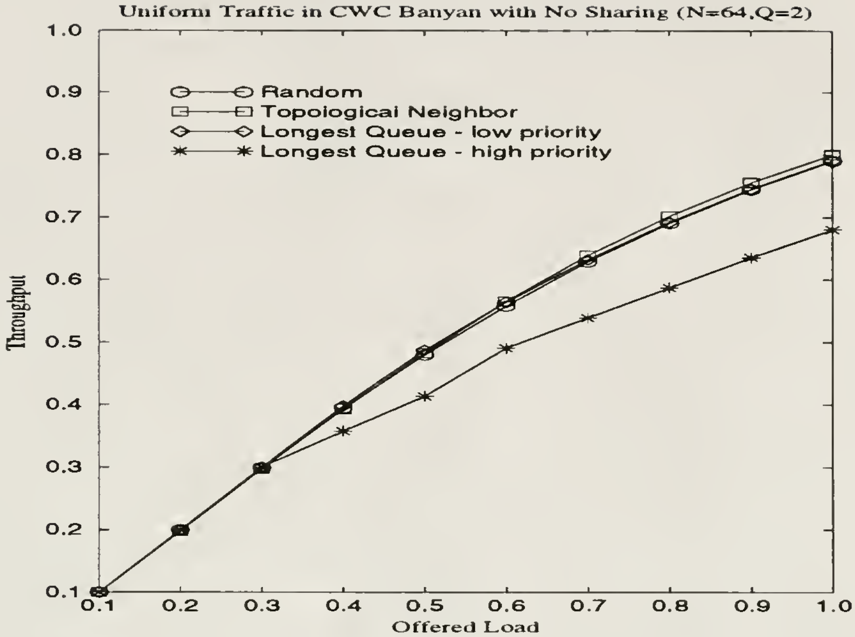

Figure 7.9. Throughput of CWC scheme under different alternate queue selections

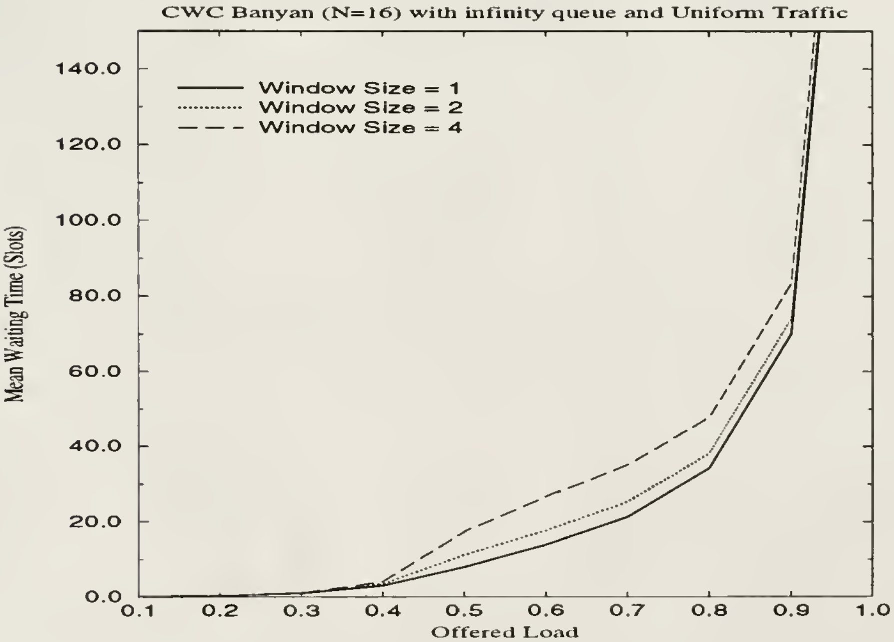

Figure 7.10. Throughput of CWC scheme under different window sizes

7.4 Chapter Summary

This Chapter presented an input queued switch architecture in which packets join subqueues according to the destination requested. The switch is then responsible for removing packets from these subqueues that will guarantee a conflict free passage. The permutation we most worked with is the sorted list, and the CWC switch then removes packets from different queues in a circular manner.

CHAPTER 8
CONCLUSION

Multistage interconnection networks (MINs) are widely used in both large-scale multiprocessors as well as in data communication networks. They provide the best tradeoff between cost and efficiency. Their biggest drawback is that these networks are blocking and may not be able to establish all free input-output pairs due to internal link conflicts.

The network delay, in addition to memory access delays, is largely the cause for performance deterioration in large-scale multiprocessors. MIN performance is adequate under regular traffic, but can be adversely affected by memory references which are non-uniform. In particular, this thesis addresses the Consecutive Requests (CR) traffic pattern, as is exhibited by vector accesses. The cause was identified as link underutilization for a long period of time. To improve the performance two strategies were adopted: the Dynamic Priority and the Bit-Reverse. The Dynamic Priority allows rapid passage of CR traffic, while the Bit-Reverse mapping scheme distributed the accesses of a unit stride access. A generalized mapping scheme which takes into consideration the access stride was developed and evaluated for non-unit stride accesses. It was shown that the generalized mapping scheme in conjunction with skewed storage improved the network performance considerably. Furthermore, the behavior of the entire system was also evaluated when the CR traffic pattern interacted with regular traffic, i.e. scalar references. Bit-Reverse mapping alone does not improve the performance dramatically, but in combination with Dynamic Priority can yield even better results than Dynamic Priority alone.

To improve the performance of MINs in data communication networks, we have first separated the requirements depending on the switching technique used in the networks. In circuit-switched networks, the objective is to grant as many calls during a particular setup period, with the least amount of overhead as possible. In packet-switched networks, the targets are to maximize packet throughput and to minimize packet loss.

The drawback to the naive parallel path setup scheme was that conflicts were resolved arbitrarily. In the Re-Attempt parallel path setup mechanism, connections which lose during a link insertion conflict in an intermediate stage get a second chance at connecting, provided the call that won the earlier arbitration was unsuccessful. The Re-Attempt scheme clearly improves on the average number of connections which can be granted and approaches the optimum connections setup within 3%.

The Circular Window Control (CWC) scheme is a simple and efficient traffic regulation scheme for input queueing switch fabrics. It is clear that the scheme is independent of the switching fabric. The queues must be synchronized in terms of the choice of primary subqueues, but no interaction is needed during the transmission. The mechanism to identify the packet destination must be incorporated such that the arriving packets can join the corresponding subqueues. The performance gain of the CWC scheme over the FIFO input queueing system is significant. With moderate queue size, the throughput can be close to that of the output queueing switch system. The mean waiting time of the CWC scheme is much less than that of the FIFO input queueing systems. This achievement is based on the facts that, if the load is high, the packets transmitted are chosen from the primary subqueues and cause no switch contention, whereas when the load is low, the selection of packets from alternate subqueues can utilize the available switch bandwidth. The most interesting observation is that the CWC scheme can make a UPP blocking switch behave just like a non-blocking switch and experience minimal switch contentions.

In this thesis, we have established that traffic patterns can have a catastrophical impact on the performance of MINs. By identifying the underlying cause of the deterioration, means may be undertaken to alleviate the symptoms. Caution should be taken, however, such that the introduced methods will not affect regular traffic adversely. Traffic patterns result from the interaction of parallel programs, and as this information becomes readily available, similar observations can be made such as was done for the CR traffic pattern. The arbitration method in the ReAPPS scheme is basically still random in the early stages because there is not enough information (such as collision count) to make an informed decision. Future work could provide improved arbitration methods. The CWC scheme is limited to a circularly sorted input pattern, which is one of the permutations which is passable in a UPP network. As a future enhancement, it may be useful to specify other permutations that are passable and adapt the CWC scheme to schedule packets in such a way that these permutations are presented to the UPP network.

REFERENCES

[ARAK90] Arakawa, N., A. Noiri, and H. Inoue, "ATM Switch for Multi-Media Switching System," *ISS*, Vol. 5, 1990, pp. 9-14.

[ARMB86] Armbruster, H., "Applications of Future Broad-band Services in the Office and Home," *IEEE Selected Areas in Communications*, Vol. 4, No. 4, July 1986.

[BAIL87] Bailey, D. H., "Vector Computer Memory Bank Contention," *IEEE Transactions on Computers*, C-36, (March 1987), pp. 293-298.

[BATC76] Batcher, K. E., "The Flip Network in STARAN," *Proceedings of the 1976 International Conference on Parallel Processing*, 1976, pp. 65-71.

[BING88] Bingham, B., and H. Bussey, "Reservation-Based Contention Resolution Mechanism for Batcher-Banyan Packet Switches," *Electronic Letters* Vol. 24, No. 13, (June 1988), pp. 772-773.

[BUDN71] Budnik, P., and D. J. Kuck, "The Organization and Use of Parallel Memories," *IEEE Transactions on Computers*, C-20, 12(December 1971), pp. 1566-1569.

[CCIT90] CCITT Study Group XVIII, Revised Draft Recommendation I.211, Geneva, Switzerland, May 9-25, 1990.

[CHEN91] Chen, J., and T. Stern, "Throughput Analysis, Optimal Buffer Allocation, and Traffic Imbalance Study of a Generic Nonblocking Packet Switch," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 3, (April 1991).

[CHEU86] Cheung, T., and J. E. Smith, "A Simulation Study of the CRAY X-MP Memory System," *IEEE Transactions on Computers*, C-35, (July, 1986), pp. 613-622.

[CISN90] Cisneros, A., "Large Packet Switch and Contention Resolution Device," *ISS*, Vol. 3, 1990, pp. 77-83.

[DIAS81] Dias, D.M., and J.R. Jump, "Analysis and Simulation of Buffered Delta Networks," *IEEE Transactions on Computers*, C-30, (April 1981), pp. 273-282.

[DIAS89] Dias, D.M., and M. Kumar, "Preventing Congestion in Multistage Networks in the Presence of Hotspots," *Proceedings 1989 International Conference on Parallel Processing*, (August 1989), pp. I.9-13.

[EVAN78] Evans, J. R., and J. J. Jarvis, "Network Topology and Integral Multicommodity Flow Problems," *Networks*, Vol. 8, 1978, pp. 107-119.

[EVEN79] Even, S., "Graph Algorithms," Computer Science Press, Rockville, Maryland, 1979.

[FORD62] Ford, L. R., Jr. and D. R. Fulkerson, "Flows in Networks," Princeton University Press, Princeton, New Jersey, 1962.

[GAJS83] Gajski, D., D. Kuck, D. Lawrie, and A. Sameh, "Cedar - A Large Scale Multiprocessor," *Proceedings 1983 International Conference on Parallel Processing*, August 1983, pp. 524-529.

[GALL91] Gallivan, K., W. Jalby, S. Turner, A. Veidenbaum, and H. Wijshoff, "Preliminary Basic Performance Analysis of the Cedar Multiprocessor Memory System," *Center for Supercomputing Research and Development Technical Report*, No. 1116, May 1991.

[GARE79] Garey, M. R. and D. S. Johnson, "Computers and Intractability - A Guide to the Theory of NP-Completeness", W. H. Freeman and Company, New York 1979.

[GOLU89] Golub, G. H., and C. F. van Loan, "Matrix Computations", The Johns Hopkins University Press, Baltimore 1989.

[GOTT83] Gottlieb, A., R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The NYU Ultracomputer - Designing a MIMD Shared Memory Parallel Machine," *IEEE Transactions on Computers*, Vol. C-32, February 1983, pp. 175-189.

[HARP87] Harper III, D. T., and J. R. Jump, "Vector Access Performance in Parallel Memories Using a Skewed Storage Scheme," *IEEE Transactions on Computers*, C-36, 12(December 1987) pp. 1440-1449.

[HARP89] Harper III, D. T., and D. A. Linebarger, "A Dynamic Storage Scheme for Conflict-Free Vector Access," *Proceedings of the International Symposium on Computer Architecture*, 1989, pp. 72-77.

[HLUC88] Hluchyj, M. G., and M. J. Karol, "Queueing in High-Performance Packet Switching," *IEEE Journal on Selected Areas in Communication* Vol. 6, No. 9 (December, 1988), pp. 1587-1597.

[HO89] Ho, W.S., and D.L. Eager, "A Novel Strategy for Controlling Hot Spot Congestion," *Proceedings 1989 International Conference on Parallel Processing*, August 1989 pp. I.14-18.

[HUAN84] Huang, A., and S. Knauer, “Starlite: A Wideband Digital Switch,” *IEEE Proceedings of 1984 Globecom*, pp. 121-125.

[HUI87] Hui, J., and E. Arthurs, “A Broadband Packet Switch for Integrated Transport,” *IEEE Journal on Selected Areas in Communication* Vol. 5, October 1987, pp. 1264-1273.

[HUI91] Hui, J., “Switching Integrated Broadband Services by Sort-Banyan Networks,” *Proceedings of the IEEE*, Vol. 79, No. 2, February 1991, pp. 145-154.

[ILIA90] Iliadis, I., and W. Denzel, “Performance of Packet Switches with Input and Output Queueing,” *Proc. ICC/SUPERCOM 1990*, April 1990.

[KARO87] Karol, M. J., M. G. Hluchyj, and S. P. Morgan, “Input Versus Output Queueing on a Space-Division Packet Switch,” *IEEE Transactions on Communications* Vol. 35, No. 12 (December 1987), pp. 1347-1356.

[KARP75] Karp, R. M., “On the Computational Complexity of Combinatorial Problems,” *Networks*, Vol. 5, 1975, pp. 45-68.

[KENN78] Kennington, J. L., “A Survey of Linear Cost Multicommodity Network Flows,” *Operations Research* Vol. 26, No. 2, March-April 1978, pp. 209-236.

[KIM90a] Kim, H.S., and A. Leon-Garcia, “A Self-Routing Multistage Switching Network for Broadband ISDN,” *IEEE Journal on Selected Areas in Communication*, Vol. 8, No. 3, April 1990, pp. 459-466.

[KIM90b] Kim, H.S., and A. Leon-Garcia, “Performance of Buffered Banyan Networks Under Nonuniform Traffic Patterns,” *IEEE Transaction on Communications*, Vol. 38, No. 5, May 1990, pp. 648-658.

[KRUS83] Kruskal, C.P., and M. Snir, “The Performance of Multistage Interconnection Networks for Multiprocessors,” *IEEE Transactions on Computers*, C-33 12(December 1983) pp. 1091-1098.

[KUMA84] Kumar, M., and J.R. Jump, “Performance Enhancement in Buffered Delta Networks Using Crossbar Switches and Multiple Links,” *Journal of Parallel and Distributed Computing*, Vol. 1, No. 1, 1984, pp. 81-103.

[LAWR75] Lawrie, D.H., “Access and Alignment of Data in an Array Processor,” *IEEE Transactions on Computers*, C-24, 12(December 1975) pp. 1145-1155.

[LEA87] Lea, C.-T., “An Alternate Path Self-Routing Packet Switching Node Having Fault Detection Capabilities,” *U.S. Patent* 4-679-186, July 1987.

[LEA92] Lea, C.-T., "New Performance Bounds of a Class of Self-Routing Networks," *IEEE Transactions on Communications*, Vol. 40, No. 3, March 1992, pp. 636-641.

[LEE91] Lee, M. J., and S.-Q. Li, "Performance of a Nonblocking Space-Division Packet Switch in a Time Variant Nonuniform Traffic Environment," *IEEE Transactions on Communications* Vol. 39, No. 10 (October 1991), pp. 1515-1524.

[MATS85] Matsumoto, K., T. Nishizeki, and N. Saito, "An Efficient Algorithm for Finding Multicommodity Flows in Planar Networks," *SIAM Journal on Computing* Vol. 14, No. 2, May 1985, pp. 289-302.

[MINZ89] Minzer, S., "Broadband ISDN and Asynchronous Transfer Mode (ATM)," *IEEE Communications Magazine*, Vol. 27, No. 9, pp. 17-24, September 1989.

[NAGA89] Nagamochi, H., and T. Ibaraki, "On Max-Flow Min-Cut and Integral Flow Properties for Multicommodity Flows in Directed Networks," *Information Processing Letters*, Vol. 31, June 1989, pp. 279-285.

[OED85] Oed, W., and O. Lange, "On the Effective Bandwidth of Interleaved Memories in Vector Processor Systems," *IEEE Transactions on Computers*, Vol. C-34, 10(October 1985), pp. 949-957.

[ORUÇ91] Oruç, A. Y., and M. Mittal, "Setup Algorithms for Cube-Connected Parallel Computers Using Recursive Karnaugh Maps," *IEEE Transactions on Computers*, Vol. C-40, 2(February 1991), pp. 217-221.

[PARK80] Parker, D. S., "Notes on Shuffle/Exchange-type Switching Networks," *IEEE Transactions on Computers*, C-29, 3(March 1980) pp. 213-222.

[PATE81] Patel, J.H., "Performance of Processor-Memory Interconnections for Multiprocessors," *IEEE Transactions on Computers*, C-30, 10(October 1981) pp. 771-780.

[PATT88] Pattavina, A., "Multichannel Bandwidth Allocation in a Broadband Packet Switch," *IEEE Journal of Selected Areas in Communications*, Vol. 6, 9(December 1988) pp. 1489-1499.

[PATT93] Pattavina, A., "Nonblocking Architectures for ATM Switching," *IEEE Communications Magazine*, 2(February 1993) pp. 38-48.

[PFIS85a] Pfister, G., W. Brantley, D. George, S. Harvey, W. Kleinfeider, K. McAuliffe, E. Melton, V. Norton, and J. Weiss, "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture," *Proceedings 1985 International Conference on Parallel Processing*, August 1985, pp. 764-771.

[PFIS85b] Pfister, G.F., and V. A. Norton, " 'Hot-Spot' Contention and Combining in Multistage Interconnection Networks," *IEEE Transactions on Computers* Vol. C-34, No. 10, October 1985, pp. 943-948.

[RAGH91] Raghavan, R., and J. P. Hayes, "Scalar-Vector Memory Interference in Vector Computers," *Proceedings 1991 International Conference on Parallel Processing*, August 1991, pp. I.180-187.

[SCOT89] Scott, S.L., and G.S. Sohi, "Using Feedback to Control Tree Saturation in Multistage Interconnection Networks," *Proceedings 16th Annual International Symposium Computer Architecture* May 1989, pp. 167-176.

[SHAP78] Shapiro, H. D., "Theoretical Limitations of the Efficient Use of Parallel Memories," *IEEE Transactions on Computers*, C-27, 5(May 1978), pp. 421-428.

[SHEN89] Shen, K.-H., "Preventing Performance Degradation of Multistage Interconnection Networks Under Two Traffic Patterns," Master's thesis, CIS Department, University of Florida, Fall, 1989.

[SIEG85] Siegel, H. J., "Interconnection Networks for Large-Scale Parallel Processing," Lexington Books, D.C. Heath and Company, Lexington, Massachusetts, 1985.

[TANG64] Tang, D. T., "Bi-Path Networks and Multicommodity Flows," *IEEE Transactions on Circuit Theory* Vol. CT-11, 1964, pp. 468-474.

[TOBA90] Tobagi, F. A., "Fast Packet Switch Architectures for Broadband Integrated Services Networks," *Proceedings IEEE* Vol. 78, No. 1, January 1990, pp. 133-167.

[TOBA91] Tobagi, F. A., T. Kwok, and F. M. Chiussi, "Architecture, Performance and Implementation of the Tandem Banyan Fast Packet Switch," *IEEE Journal on Selected Areas in Communications,* Vol. 9, October 1991, pp. 1173-1193.

[TURN89] Turner, S.W., and A. Veidenbaum, "Burst Traffic in MIN-Based Shared-Memory Systems," *Center for Supercomputing Research and Development Technical Report*, No. 855, February 1989.

[WIJS85] Wijshoff, H., and J. van Leeuwen, "The Structure of Periodic Storage Schemes for Parallel Memories," *IEEE Transactions on Computers*, C-34, 6(June 1985), pp. 501-505.

[WOOD90] Woodruff, G. M., and R. Kositpaiboon, "Multimedia Traffic Management Principles for Guaranteed ATM Network Performance," *IEEE Journal on Selected Areas in Communications,* Vol. 8, No. 3, April 1990, pp. 437-446.

[WU80] Wu, C.-L., and T.-Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Transactions on Computers,* Vol. C-29, pp. 694-702, August 1980.

[YEH87] Yeh, Y., M. Hluchyj, and A. Acampora, "The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching," *IEEE Journal on Selected Areas in Communication* Vol. 5, October 1987, pp. 1274-1283.

[YEW87] Yew, P.-C., N.-F. Tzeng, and D. H. Lawrie, "Distributing Hot-Spot Addressing in Large-Scale Multiprocessors," *IEEE Transactions on Computers,* Vol. C-36, No. 4, April 1987, pp. 388-395.

[ZEGU93] Zegura, E. W., "Architectures for ATM Switching Systems," *IEEE Communications Magazine,* 2(February 1993) pp. 28-37.

BIOGRAPHICAL SKETCH

The author was born in San Nicolas, Aruba, in 1967, where she attended the Voortgezet Wetenschappelijk Onderwijs of the Colegio Arubano. She obtained a Bachelor of Science degree in computer science at the Florida Institute of Technology in 1988. It was at the Florida Institute of Technology where she developed a deep esteem and respect for research and investigation. With the intention of pursuing a graduate degree, she enrolled at the University of Florida in 1988. In 1990 she changed her major field of interest from computer vision to performance evaluation and has been working toward a doctorate since then. During all those years she served as a teaching or a research assistant in the Computer and Information Sciences Department. While at the University of Florida she became active in the creation of the Graduate Student Organization in the Computer and Information Sciences Department. During her last years at the University she also became involved with the Graduate Assistants United of which she was the treasurer. Her research interests include high-speed networks, performance evaluation, parallel and distributed systems.

It is the author's firm belief that the road to success is paved with personal tragedies and sacrifices. Too often the long term goals obscure what is indeed important: the small moments in life.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yann-Hang Lee

Yann-Hang Lee, Chair
Associate Professor of Computer and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Yuan-Chieh Chow

Yuan-Chieh Chow
Professor of Computer and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Richard Newman-Wolfe
Assistant Professor of Computer and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Tim Davis

Timothy Davis
Assistant Professor of Computer and
Information Sciences

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Sherman Bai

Sherman X. Bai
Assistant Professor of Industrial and
Systems Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

August 1993

for Winfred M. Phillips
Dean, College of Engineering

Dean, Graduate School